

C#.NET 8.0

Harsha Vardhan

.NET Expert

.NET Fundamentals.....	26
Introduction to .NET	26
What is .NET?	26
Parts of .NET	26
History of .NET.....	27
History of .NET.....	27
Versions of .NET	29
CIL and CLR	30
Execution Model of .NET Programs	30
CIL (Common Intermediate Language) or MSIL.....	31
CLR (Common Language Runtime).....	31
Sub Components of CLR	32
.NET Framework Architecture.....	34
What is .NET Framework.....	34
System Requirements of .NET Framework:	34
.NET Framework Architecture.....	35
CLS and CTS	36
CLS (Common Language Specification)	36
CTS (Common Type System)	36
FCL and BCL	37
FCL (Framework Class Library)	37
Introduction to Visual Studio	38
What is Visual Studio.....	38
Versions of Visual Studio.....	38
System Requirements of Visual Studio 2019:	39
Project	40
Solution	40
File Types of .NET	40
Folder Structure of .NET Project in Visual Studio	41

Visual Studio - Installation.....	42
Steps to install “Visual Studio 2019”:	42
C#.NET – Language Fundamentals.....	55
Introduction to C#.NET.....	55
What is C#.NET:.....	55
Versions of C#.NET	56
Tokens of C#.NET.....	56
Naming Conventions for Identifiers	57
Identifier Naming Rules.....	57
Console Applications - Introduction.....	58
What is Console Application?	58
Rules of Console Application.....	58
Console Application - Example.....	59
The “System.Console” class	60
1. System.Console.WriteLine().....	60
2. System.Console.ReadLine()	61
3. System.Console.ReadKey()	61
4. System.Console.ReadLine()	61
5. System.Console.Clear()	61
System.Console.WriteLine – Example.....	62
System.Console.ReadKey – Example	63
System.Console.Clear - Example	65
Variables.....	68
Variables - Example	69
Data Types.....	71
Types of Data types	71
Numerical Data Types	71
Default Numerical data types	72
Non-Numerical Data Types	73
Numerical Data Types - Example	73
MinValue and MaxValue	76

MinValue and MaxValue - Example	76
char - Example.....	78
String	79
String - Example.....	80
Bool - Example.....	81
Operators	83
Arithmetical Operators	83
Arithmetical Operators - Example.....	84
Assignment Operators.....	86
Assignment Operators - Example.....	86
Increment / Decrement Operators	88
Increment / Decrement Operators - Example	88
Relational Operators	90
Relational Operators - Example	90
Logical Operators	92
Logical Operators - Example.....	93
Concatenation Operator	94
Concatenation Operator - Example	95
Conditional Operator	96
Conditional Operator - Example.....	96
Control Statements	98
Types of control statements:	98
If.....	99
If - Example.....	101
If – Else - Example.....	102
Else – If - Example.....	104
Nested If - Example	105
Switch – Case.....	107
Switch – Case - Example.....	107
While	109
While - Example.....	109

Do – While	111
Do – While - Example	111
for	113
for - Example	113
Break.....	115
Break - Example.....	115
Continue	116
Continue - Example	117
Goto.....	118
Goto - Example	118
Nested For Loops.....	120
Nested For Loops - Example.....	120
C#.NET – Object Oriented Programming (OOP).....	123
Introduction to Object Oriented Programming	123
Types of Programming Languages	123
Introduction of Object Oriented Programming (OOP).....	124
Object	124
Fields.....	125
Methods	125
Class.....	126
Reference variables.....	128
Syntax of creating class	128
Syntax of creating a Field in a class:.....	128
Syntax of creating a method in a class:.....	128
Syntax of creating reference variable	129
Syntax of creating object (in heap):	129
Memory allocation of Objects.....	129
Principles of Object Oriented Programming	130
Principles of OOP.....	130
Encapsulation:	130
Abstraction:	130

Inheritance:	130
Polymorphism:	131
Types of polymorphism:.....	131
Access Modifiers	132
Access Modifiers	132
Access Modifiers - Example.....	135
Access Modifiers for classes.....	138
Access Modifiers for Classes	138
Object Oriented Programming – Example	138
Object Oriented Programming – Student Example	140
Object Oriented Programming – Employee Example	142
Program.cs.....	143
Static Fields.....	145
Static Fields - Example.....	146
Constant Fields.....	149
Constant Fields - Example	149
ReadOnly Fields.....	151
ReadOnly Fields - Example	151
Methods	153
Methods – Simple Example.....	154
Methods – Arguments and Return Example	155
Scope of Variables	157
Methods – Example - Age	158
Methods – Example 2 - Numbers.....	160
Methods – Example 3 - Login	163
Methods – Example 4 - Student.....	165
"this" keyword.....	170
"this" keyword - Example	170
Static Methods	172
Static Methods - Example	173
Reference Variables as Arguments	175

Reference Variables as Arguments - Example	175
Reference Variables as Fields.....	177
Reference Variables as Fields - Example	177
Default Arguments	179
Default Arguments - Example	179
Named Parameters	181
Named Parameters - Example.....	181
Methods Overloading.....	183
Methods Overloading - Example.....	183
Types of Parameters.....	185
Call by value - Example.....	186
Call by reference - Example.....	187
Call by output - Example	189
Type Conversion.....	191
Implicit Casting - Example	191
Explicit Casting (or) Type Casting	193
Explicit Casting (or) Type Casting - Example	193
Parsing	195
Parsing - Example	195
TryParse.....	197
.TryParse - Example	197
.TryParse – Example 2	199
Output	200
Conversion Methods	201
Conversion Methods - Example	202
Constructors	204
Types of Constructor	204
Default Constructor.....	205
Parameterless Constructor - Example.....	205
Parameterized Constructor - Example	207
Constructors - Student - Example	209

Object Initializer	212
Object Initializer - Example	212
"Set" and "Get" Methods.....	216
Set Method.....	216
Get Method	216
"Set" and "Get" Methods - Example	217
Properties	219
Properties - Example	221
Readonly and WriteOnly Properties	223
Readonly Properties - Example	224
Automatic Properties	226
Automatic Properties - Example	226
Inheritance	228
Inheritance - Example	228
Inheritance – Example 2.....	231
"base" keyword.....	234
"base" keyword - Example	234
Parent class's constructor	237
Parent class's constructor - Example	237
Method Hiding	241
Method Hiding - Example.....	242
Method Hiding – Example 2	244
Method Overriding.....	247
Normal Methods (vs) Virtual Methods	247
Method Hiding (vs) Method Overriding.....	248
Method Overriding - Example	249
Method Overriding – Example 2	251
Abstract Classes.....	254
Abstract Classes - Example.....	255
Abstract Classes – Example 2	257
Abstract Methods.....	260

Abstract Methods - Example	262
Abstract Methods – Example 2	264
Interfaces.....	268
Abstract Class (vs) Interface	268
Interfaces - Example	271
Interfaces – Example 2	273
Dynamic Polymorphism	276
Dynamic Polymorphism - Example.....	277
Dynamic Polymorphism – Example 2.....	279
Multiple Inheritance using Interfaces	282
Multiple Inheritance using Interfaces - Example	282
Interface Inheritance.....	286
Interface Inheritance - Example.....	286
Sealed Classes.....	290
Sealed Classes - Example.....	290
Namespaces	293
Namespaces - Example	293
Child Namespaces	296
Child Namespaces - Example	296
Child Namespaces – Example 2.....	298
“using” statement	301
“using” statement - Example	301
“using - alias” statement.....	303
“using - alias” statement - Example	303
Creating Separate Files for Classes	306
Creating Separate Files for Classes - Example.....	306
Partial Classes.....	309
Partial Classes - Example	309
Enumerations	313
Enumerations - Example	313
Structures	316

Structures - Example	318
Structures with Constructors - Example	320
Standard Data Types	323
The "System.Object" class.....	324
“System.Object” Class - Example	324
Methods of "System.Object" class.....	326
Methods of “System.Object” class - Example.....	327
Boxing	330
Boxing - Example	330
Unboxing	332
Unboxing - Example	332
Static Constructors	334
Static Constructors - Example	335
Static Classes	338
Static Classes - Example	338
Generic Classes.....	341
Generic Classes - Example	341
Multiple Generics - Example	343
Arrays	346
Arrays - Example.....	346
Arrays with For Loop	348
Arrays with For Loop - Example	348
Arrays with Foreach Loop in C#.NET	350
Arrays with Foreach Loop – Example	350
“System.Array” class	353
“System.Array.IndexOf” method	354
“System.Array.IndexOf” method - Example.....	354
“System.Array.IndexOf” method – with NotFound - Example	356
“System.Array.BinarySearch” method	358
“System.Array.BinarySearch” method - Example	358
“System.Array.Clear” method.....	360

"System.Array.Clear" method - Example	360
"System.Array.Resize" method	363
"System.Array.Resize" method - Example	363
"System.Array.Sort" method	366
"System.Array.Sort" method - Example.....	366
"System.Array.Reverse" method	368
"System.Array.Reverse" method - Example	368
"System.Array.CopyTo" method	370
"System.Array.CopyTo" method - Example	370
Multi-Dimensional Array	373
Multi-Dimensional Array - Example	373
Collections	376
List of collections classes in .NET:	376
Arrays (vs) Collections	376
The "List" class.....	377
The "List" class - Example.....	377
The "List" class – with “for” loop	380
The "List" class – with “for” loop - Example.....	380
The "List" class – with “foreach” loop.....	382
The "List" class – with “foreach” loop - Example	382
"System.Collections.Generic. List.Add" method	384
"System.Collections.Generic. List.Add" method - Example	384
"System.Collections.Generic. List.Insert" method	386
"System.Collections.Generic.List.Insert" method - Example	386
"System.Collections.Generic. List.AddRange" method.....	388
"System.Collections.Generic. List.AddRange" method - Example	388
"System.Collections.Generic. List.InsertRange" method	391
"System.Collections.Generic. List.InsertRange" method - Example	391
"System.Collections.Generic. List.Remove" method in C#.NET.....	394
"System.Collections.Generic. List.Remove" method - Example	394
"System.Collections.Generic. List.RemoveAt" method	396

"System.Collections.Generic. List.RemoveAt" method - Example.....	396
"System.Collections.Generic. List.Clear" method	399
"System.Collections.Generic. List.Clear" method - Example	399
"System.Collections.Generic. List.IndexOf" method	401
"System.Collections.Generic. List.IndexOf" method - Example.....	401
"System.Collections.Generic. List.IndexOf" method – Not Found- Example.....	403
"System.Collections.Generic. List.BinarySearch" method	405
"System.Collections.Generic. List.BinarySearch" method - Example	406
"System.Collections.Generic. List.Contains" method	407
"System.Collections.Generic. List.Contains" method - Example	407
"System.Collections.Generic. List.Reverse" method	409
"System.Collections.Generic. List.Reverse" method - Example.....	409
"System.Collections.Generic. List.Sort" method.....	411
"System.Collections.Generic. List.Sort" method - Example.....	411
System.Collections.Generic. List - Sort Descending - Example	413
Collection Filter	415
Collection Filter - Example	415
LINQ.....	417
LINQ - Example	417
Collection of Objects	420
Collection of Objects - Example	420
Collection of Objects – Filter - Example	422
Collection of Objects – LINQ - Example.....	424
The "Dictionary" class	427
The "Dictionary" class - Example.....	427
The "SortedList" class.....	429
The "SortedList" class - Example	429
The "Hashtable" class in .NET	431
The "Hashtable" class - Example	431
The "ArrayList" class in .NET	433
The "ArrayList" class - Example	433

The "typeof" operator in .NET.....	435
The "typeof" operator - Example	435
The "System.String" class (or) String Handling	439
"System.String.Length" - Example	442
"System.String.ToUpper" - Example	444
"System.String.ToLower" - Example	445
"System.String.GetChar" - Example	447
"System.String.Substring" – Example 1	449
"System.String.Substring" – Example 2	450
"System.String.Remove" – Example	451
"System.String.Remove" – Example 2	453
"System.String.Insert" – Example	455
"System.String.Equals" – Example	456
"System.StringComparison.OrdinalIgnoreCase" – Example	458
"System.String.StartsWith" – Example	459
"System.String.EndsWith" – Example	461
"System.String.Contains" – Example	462
"System.String.IndexOf" – Example 1	464
"System.String.IndexOf" – Example 2	465
"System.String.IndexOf" – Example 3	467
"System.String.LastIndexOf" – Example	468
"System.String.Replace" – Example	470
"System.String.ToCharArray" – Example	471
Converting CharArray to String - Example	474
"System.String.Split" – Example.....	475
"System.String.Trim" – Example	477
"System.String.Format" – Example.....	479
"System.String.Reverse" – Example	480
String – WordsCount - Example	482
String – Character Occurrence Count - Example.....	484
String – Alphabetic Count - Example.....	486

String – Word Occurrence Count - Example	488
String – Title Case - Example	490
String – Currency into words - Example.....	492
String – Multiple Concatenations - Example.....	496
The "System.Text.StringBuilder" class	498
The "System.Text.StringBuilder" class - Example	499
The "System.DateTime" structure (or) Date & Time Handling	501
The "System.DateTime" – First Example.....	503
The "System.DateTime.Now" –Example	505
The "System.DateTime" – Inner Values – Example.....	506
The "System.DateTime.DayOfWeek" – Example	508
The "System.DateTime.DayOfYear" – Example	510
The "System.DateTime. ToShortDateString" – Example.....	511
The "System.DateTime. ToLongDateString" – Example.....	513
The "System.DateTime. ToShortTimeString" – Example	514
The "System.DateTime. ToLongTimeString" – Example	516
The "System.DateTime" – Custom Formats – Example	517
The "System.DateTime.Subtract" – Example	519
The "System.DateTime.AddDays" – Example	521
The "System.Math" class.....	523
The "System.Math.Abs" –Example	524
The "System.Math.Floor" –Example	526
The "System.Math.Ceiling" – Example.....	527
The "System.Math.Round" – Example 1	528
The "System.Math.Round" – Example 2	530
The "System.Math.Max" –Example	531
The "System.Math.Min" – Example	533
The "System.Math.Pow" –Example	534
The "System.Math.Sqrt" –Example.....	536
"System.Diagnostics.Process" class	538
"System.Diagnostics.Process" class - Example	538

Command Line Arguments.....	540
Command Line Arguments - Example.....	540
Nullable Data Types.....	543
Nullable Data Types - Example.....	543
Exception Handling.....	545
Exception Handling - Example.....	546
Destructor	549
Destructor - Example.....	549
The "System.IDisposable" interface	552
The "System.IDisposable" interface - Example	553
Garbage Collection	555
Garbage Collection - Example	555
Delegates.....	557
Single Cast Delegates - Example.....	558
Multi Cast Delegates - Example	560
Events	562
Events - Example	563
Anonymous Methods.....	566
Anonymous Methods - Example	566
Lambda Expressions	569
Lambda Expressions - Example	569
Inline Lambda Expressions	572
Inline Lambda Expressions - Example	572
Inner Classes.....	575
Inner Classes - Example	575
Indexer.....	578
Indexer - Example.....	579
Assemblies.....	581
Class Libraries & Private Assemblies - Example	584
Shared Assemblies - Example.....	588
Documentation Comments	593

Documentation Comments - Example	593
Extension Methods	596
Extension Methods - Example.....	597
C# – Console - System.IO Namespace	601
Introduction to "System.IO" namespace	601
The "System.IO.FileInfo" class	602
The "System.IO.FileInfo" class - Example.....	605
The "System.IO.DirectoryInfo" class	607
The "System.IO.DirectoryInfo" class - Example	610
The "System.IO.Directory" class	613
The "System.IO.Directory" class - Example.....	614
The "System.IO.File" class.....	617
The "System.IO.File" class - Example	618
The "System.IO.FileStream" class	621
The "System.IO.StreamWriter" class	623
The "System.IO.StreamWriter" class - Example	624
The "System.IO.StreamReader" class	626
The "System.IO.StreamReader" class - Example	627
C#.NET – Console – ADO.NET	629
Database Basics.....	629
Introduction to ADO.NET	631
List of pre-defined classes in ADO.NET	631
ADO.NET – "SqlConnection" class.....	632
SqlConnection – Windows Authentication – Example.....	636
SqlConnection – SQL Server Authentication – Example	639
The "SqlCommand" class in ADO.NET	642
ADO.NET - ExecuteScalar	646
SqlCommand – ExecuteScalar – Example	647
SqlCommand – ExecuteScalar – Example 2	650
ADO.NET – Connection Oriented Model – Introduction.....	653

The "SqlDataReader" class in ADO.NET	654
ADO.NET – Connection Oriented Model.....	656
ADO.NET Connection Oriented Model – Single Record – Example	657
ADO.NET Connection Oriented Model – Multiple Records – Example.....	661
The "SqlParameter" class in ADO.NET	665
ADO.NET Connection Oriented Model – SqlParameter – Example	668
ADO.NET Disconnected Model.....	672
The “SqlDataAdapter” class	673
The “DataSet” class	674
DataSet - Example	674
ADO.NET Disconnected Model – Example	679
ADO.NET Disconnected Model – Multiple Tables - Example.....	683
ADO.NET Disconnected Model – Joins - Example	688
ADO.NET Non Query - Insertion – Example	692
ADO.NET Non-Query - Updation – Example	695
ADO.NET Non-Query - Deletion – Example.....	698
ADO.NET Non Query – Insertion – With SqlParameter – Example.....	701
ADO.NET Non-Query – Updation – With SqlParameter – Example.....	705
ADO.NET Non-Query – Deletion – With SqlParameter – Example	709
Stored Procedures Calling in ADO.NET	712
ADO.NET Non Query – Insertion – With Stored Procedures – Example.....	713
ADO.NET Non Query – Updation – With Stored Procedures – Example	717
ADO.NET Non Query – Deletion – With Stored Procedures – Example	721
ADO.NET - Transactions	725
ADO.NET – Transactions - Example.....	726
The "OleDb" namespace in ADO.NET	730
Connection Strings in ADO.NET.....	731
ADO.NET – Oracle - Example.....	732
ADO.NET – MS Access - Example	743
ADO.NET – MS Excel - Example	755
ADO.NET – SQL Server to Oracle - Example	764

ADO.NET – SQL Server to MS Excel - Example	771
ADO.NET – Oracle to SQL Server - Example	779
ADO.NET – Excel to SQL Server - Example	786
ADO.NET – SQL Server to File - Example	794
ADO.NET – File to SQL Server - Example	799
The "SqlCommandBuilder" class in ADO.NET	804
ADO.NET – SqlCommandBuilder – DataSet – Insertion - Example	805
ADO.NET – SqlCommandBuilder – DataSet – Updation - Example.....	809
ADO.NET – SqlCommandBuilder – DataSet – Deletion - Example.....	813
N-Tier Architecture.....	817
N-Tier Architecture - Example.....	817
LINQ to SQL	825
LINQ to SQL - Example.....	826
C#.NET – Console – Entity Framework.....	831
Intro to ADO.NET Entity Framework	831
Entity Framework - Example	834
Entity Framework – FirstOrDefault - Example	839
Entity Framework – Insertion - Example.....	844
Entity Framework – Updation - Example	849
Entity Framework – Deletion - Example	854
C#.NET – Windows Forms Applications.....	858
Introduction to Windows Forms Applications	858
The “System.Windows.Forms.Form” class	859
Programming Model of Windows Form.....	860
Windows Forms Application – First Example.....	861
Form Constructor - Example	863
Properties of “System.Windows.Forms.Form” class	865
“System.Windows.Forms.Form. Text” property - Example	867
“System.Windows.Forms.Form. ShowIcon” property - Example	869
“System.Windows.Forms.Form. ShowInTaskBar” property -Example	871

“System.Windows.Forms.Form. MinimizeBox” property - Example	872
“System.Windows.Forms.Form. MaximizeBox” property - Example	874
“System.Windows.Forms.Form. ControlBox” property - Example	876
“System.Windows.Forms.Form. TopMost” property - Example	877
“System.Windows.Forms.Form. WindowState” property –Example	879
“System.Windows.Forms.Form. FormBorderStyle” property – Example.....	880
“System.Windows.Forms.Form. Cursor” property - Example	882
“System.Windows.Forms.Form. BackColor” property - Example.....	884
“System.Windows.Forms.Form. BackgroundImage” property – Example	885
“System.Windows.Forms.Form. Size” property - Example.....	887
“System.Windows.Forms.Form. Location” property - Example	889
“System.Windows.Forms.Form. Icon” property - Example	890
Events of “System.Windows.Forms.Form” class	892
“System.Windows.Forms.Form. Load” event - Example	893
“System.Windows.Forms.Form. Shown” event - Example	896
“System.Windows.Forms.Form. Click” event - Example	897
“System.Windows.Forms.Form. DoubleClick” event - Example.....	899
“System.Windows.Forms.Form.MouseClick” event - Example	901
“System.Windows.Forms.Form. MouseMove” event - Example	903
“System.Windows.Forms.Form. KeyPress” event - Example.....	904
“System.Windows.Forms.Form. FormClosing” event - Example.....	906
“System.Windows.Forms.Form. FormClosing” event - Example 2	908
Methods of “System.Windows.Forms.Form” class.....	910
“System.Windows.Forms.Form. Hide” method - Example	911
“System.Windows.Forms.Form .Show” method - Example	913
“System.Windows.Forms.Form. Close” method - Example.....	915
Introduction to Windows Forms Controls	917
“System.Windows.Forms.Label” class	919
“System.Windows.Forms.Label” class - Example	922
“System.Windows.Forms. Button” class.....	925
“System.Windows.Forms. Button” class - Example	929

“System.Windows.Forms. Button” class – with Image - Example	932
“System.Windows.Forms. Button” class – with Label - Example	934
“System.Windows.Forms. TextBox” class	938
“System.Windows.Forms. TextBox” class - Example	943
“System.Windows.Forms. TextBox” class – With AutoComplete - Example.....	947
“System.Windows.Forms. TextBox” class – With AutoComplete – Example 2	949
“System.Windows.Forms. TextBox” class – With KeyPress – Example	952
“System.Windows.Forms. TextBox” class – With KeyPress – Example 2	955
“System.Windows.Forms. TextBox” class – With Enter and Leave – Example.....	957
“System.Windows.Forms. TextBox” class – With TabIndex – Example	961
“System.Windows.Forms. TextBox” class – Math – Example.....	965
“System.Windows.Forms. NumericUpDown” class	970
“System.Windows.Forms. NumericUpDown” class – Example	973
“System.Windows.Forms. DateTimePicker” class	976
“System.Windows.Forms. DateTimePicker” class – Example	980
“System.Windows.Forms. MonthCalendar” class	983
“System.Windows.Forms. MonthCalendar” class – Example	987
“System.Windows.Forms. ToolTip” class	991
“System.Windows.Forms. ToolTip” class – Example	993
“System.Windows.Forms. ErrorProvider” class	995
“System.Windows.Forms. ErrorProvider” class – Example	997
“System.Text. RegularExpressions. Regex” class	1000
“System.Text. RegularExpressions. Regex” class – Example.....	1001
“System.Windows.Forms. MaskedTextBox” class	1005
“System.Windows.Forms. MaskedTextBox” class - Example	1008
“System.Windows.Forms. CheckBox” class	1011
“System.Windows.Forms. CheckBox” class - Example	1014
“System.Windows.Forms. RadioButton” class.....	1016
“System.Windows.Forms. RadioButton” class - Example.....	1019
“System.Windows.Forms. ComboBox” class	1023
“System.Windows.Forms. ComboBox” class - Example	1026

“Cascading ComboBox” - Example.....	1029
“System.Windows.Forms. ListBox” class	1032
“System.Windows.Forms. ListBox” class - Example.....	1035
“System.Windows.Forms. CheckedListBox” class.....	1038
“System.Windows.Forms. CheckedListBox” class – Example	1041
“System.Windows.Forms. TreeView” class.....	1044
“System.Windows.Forms. TreeView” class - Example.....	1049
The “System.Windows.Forms. PictureBox” class	1052
“System.Windows.Forms. PictureBox” class - Example	1055
“System.Windows.Forms. PictureBox” class – Example 2.....	1057
The “System.Windows.Forms. Panel” class	1061
“System.Windows.Forms. Panel” class - Example	1063
The “System.Windows.Forms. GroupBox” class.....	1066
“System.Windows.Forms. GroupBox” class - Example.....	1069
The “System.Windows.Forms. SplitContainer” class	1072
“System.Windows.Forms. SplitContainer” class - Example	1075
The “System.Windows.Forms. TabControl” class	1077
“System.Windows.Forms. TabControl” class - Example	1080
The “System.Windows.Forms. FlowLayoutPanel” class	1083
“System.Windows.Forms. FlowLayoutPanel” class - Example	1086
“System.Windows.Forms. LinkLabel” class.....	1088
“System.Windows.Forms. LinkLabel” class - Example	1091
“System.Windows.Forms. WebBrowser” class.....	1094
“System.Windows.Forms. WebBrowser” class - Example	1096
“System.Windows.Forms.Timer” class	1099
“System.Windows.Forms. Timer” class - Example.....	1101
“System.Windows.Forms.Timer” class – with Time - Example.....	1104
“System.Windows.Forms.Timer” class – with Counter - Example.....	1106
“System.Windows.Forms.Timer” class – with Slide Show - Example	1108
“System.Windows.Forms. ProgressBar” class	1112
“System.Windows.Forms. ProgressBar” class - Example.....	1115

“System.Windows.Forms.NotifyIcon” class	1118
“System.Windows.Forms.NotifyIcon” class - Example	1120
Popup Boxes.....	1123
PopupBoxes - Example	1123
“System.Windows.Forms.ColorDialog” class	1130
“System.Windows.Forms.ColorDialog” class - Example	1131
“System.Windows.Forms.FontDialog” class	1135
“System.Windows.Forms.FontDialog” class - Example.....	1137
“System.Windows.Forms.FolderBrowserDialog” class	1140
“System.Windows.Forms.FolderBrowserDialog” class - Example	1143
“System.Windows.Forms.OpenFileDialog” class	1146
“System.Windows.Forms.OpenFileDialog” class - Example	1148
“System.Windows.Forms.SaveFileDialog” class.....	1152
“System.Windows.Forms.SaveFileDialog” class - Example.....	1154
“System.Windows.Forms.MenuStrip” class.....	1158
“System.Windows.Forms.MenuStrip” class - Example	1163
“System.Windows.Forms.ContextMenuStrip” class	1167
“System.Windows.Forms.ContextMenuStrip” class - Example	1170
“System.Windows.Forms.ToolStrip” class	1174
“System.Windows.Forms.ToolStrip” class - Example	1178
“System.Windows.Forms.StatusStrip” class	1183
“System.Windows.Forms.StatusStrip” class - Example	1187
“System.Windows.Forms.RichTextBox” class	1190
“System.Windows.Forms.RichTextBox” class - Example	1195
User Controls.....	1208
User Controls - Example	1209
Windows Forms Control Library.....	1214
Windows Forms Control Library - Example.....	1214
“System.Net.Mail.SmtpClient” class	1220
“System.Net.Mail.SmtpClient” class - Example	1221
Multi-Threading.....	1225

Multi Threading - Example	1229
Task Parallel Library.....	1231
Task Parallel Library - Example.....	1233
Windows Services.....	1236
Windows Services - Example.....	1237
C#.NET – WinForms – System.IO Namespace.....	1242
The “System.IO.FileInfo” class - Example.....	1242
The “System.IO.DirectoryInfo” class - Example	1246
The “System.IO.Directory” class - Example.....	1252
The “System.IO.File” class - Example	1255
The “System.IO.StreamWriter” class - Example	1260
The “System.IO.StreamReader” class - Example	1263
C#.NET – WinForms – ADO.NET	1267
SqlConnection – Windows Authentication – Example.....	1267
SqlConnection – SQL Server Authentication – Example	1270
SqlCommand – ExecuteScalar – Example	1274
SqlCommand – ExecuteScalar – Example 2	1277
ADO.NET Connection Oriented Model – Single Record – Example	1281
ADO.NET Connection Oriented Model – Multiple Records – Example.....	1285
ADO.NET Connection Oriented Model – Multiple Records - Label – Example	1290
ADO.NET Connection Oriented Model – SqlParameter – Example	1294
ADO.NET Connection Oriented Model – SqlParameter – ComboBox – Example.....	1299
DataSet - Example	1305
ADO.NET Disconnected Model – Example	1311
ADO.NET Disconnected Model – Multiple Tables - Example.....	1316
ADO.NET Disconnected Model – Joins - Example	1322
ADO.NET Disconnected Model – Record Navigations - Example	1327
ADO.NET Disconnected Model – ComboBox - Example	1334
ADO.NET Disconnected Model – ComboBox - DataSource - Example	1340
ADO.NET Disconnected Model – DataGridView - Example	1347

ADO.NET Disconnected Model – DataGridView - SqlCommandBuilder - Example	1350
ADO.NET Disconnected Model – Master – Child – Example	1354
ADO.NET Disconnected Model – Cascading ComboBox – Example	1360
ADO.NET Non Query - Insertion – Example	1366
ADO.NET Non Query - Updation – Example.....	1369
ADO.NET Non Query - Deletion – Example.....	1372
ADO.NET Non Query – Insertion – With SqlParameter – Example.....	1376
ADO.NET Non Query – Updation – With SqlParameter – Example	1381
ADO.NET Non Query – Deletion – With SqlParameter – Example	1386
ADO.NET Non Query – Insertion – With Stored Procedures – Example.....	1390
ADO.NET Non Query – Updation – With Stored Procedures – Example	1395
ADO.NET Non Query – Deletion – With Stored Procedures – Example	1401
ADO.NET Disconnected Model – ComboBox – Stored Procedures – Example.....	1405
ADO.NET Disconnected Model – Updation with ComboBox - Example	1411
ADO.NET – CRUD (Create, Retrieve, Update, Delete) - Example.....	1418
ADO.NET – Registration Form - Example	1432
ADO.NET – Login Form - Example	1439
ADO.NET – Transactions - Example.....	1445
ADO.NET – Oracle - Example.....	1449
ADO.NET – MS Access - Example	1461
ADO.NET – MS Excel - Example	1475
ADO.NET – SQL Server to Oracle - Example	1484
ADO.NET – SQL Server to MS Excel - Example	1491
ADO.NET – Oracle to SQL Server - Example	1500
ADO.NET – Excel to SQL Server - Example	1508
ADO.NET – SQL Server to File - Example	1516
ADO.NET – File to SQL Server - Example	1522
ADO.NET – SqlCommandBuilder – DataSet – Insertion - Example	1528
ADO.NET – SqlCommandBuilder – DataSet – Updation - Example.....	1533
ADO.NET – SqlCommandBuilder – DataSet – Deletion - Example	1538
N-Tier Architecture - Example.....	1542

LINQ to SQL - Example.....	1551
C#.NET – WinForms – Entity Framework.....	1559
Entity Framework - Example	1559
Entity Framework – Insertion - Example.....	1565
Entity Framework – Updation - Example	1571
Entity Framework – Deletion - Example	1577

HARSHA

.NET Fundamentals

Introduction to .NET

What is .NET?

- .NET is an “application development platform”, which is used to develop desktop, web and mobile applications.
- “Application” is a program (collection of instructions) that can run based on the operating system.
- “Application Development” is a process of creating new applications based on the client’s requirements.
- “Application Development Platform” is a software tool, based on which you can develop applications.
- .NET is developed by Microsoft Corporation in 2002.
- “NET” is named to mean “Internet Age” applications; it has no abbreviation.

Parts of .NET

- .NET is mainly divided into 3 parts:

1. C#.NET (C Sharp .NET)
2. ASP.NET (Active Server Pages .NET)
3. Xamarin.NET

1. C#.NET

- Covers language basics.
- It is used to develop “Stand-alone applications” that runs based on the single machine.
- Ex: Calculator, Bill Generation Applications etc.

2. ASP.NET

- Covers server side programming.
- It is used to develop “Web applications” that runs based on “Client-server architecture”.

3. Xamarin.NET

- Covers mobile programming.
- It is used to develop “Mobile applications” that runs based on “Mobile devices”.

History of .NET

History of .NET

- 1997: Microsoft wants to create a better programming language to develop standalone and client-server applications. Microsoft team (led by Anders Hejlsberg, founder of Turbo Pascal) has started developing the new language.

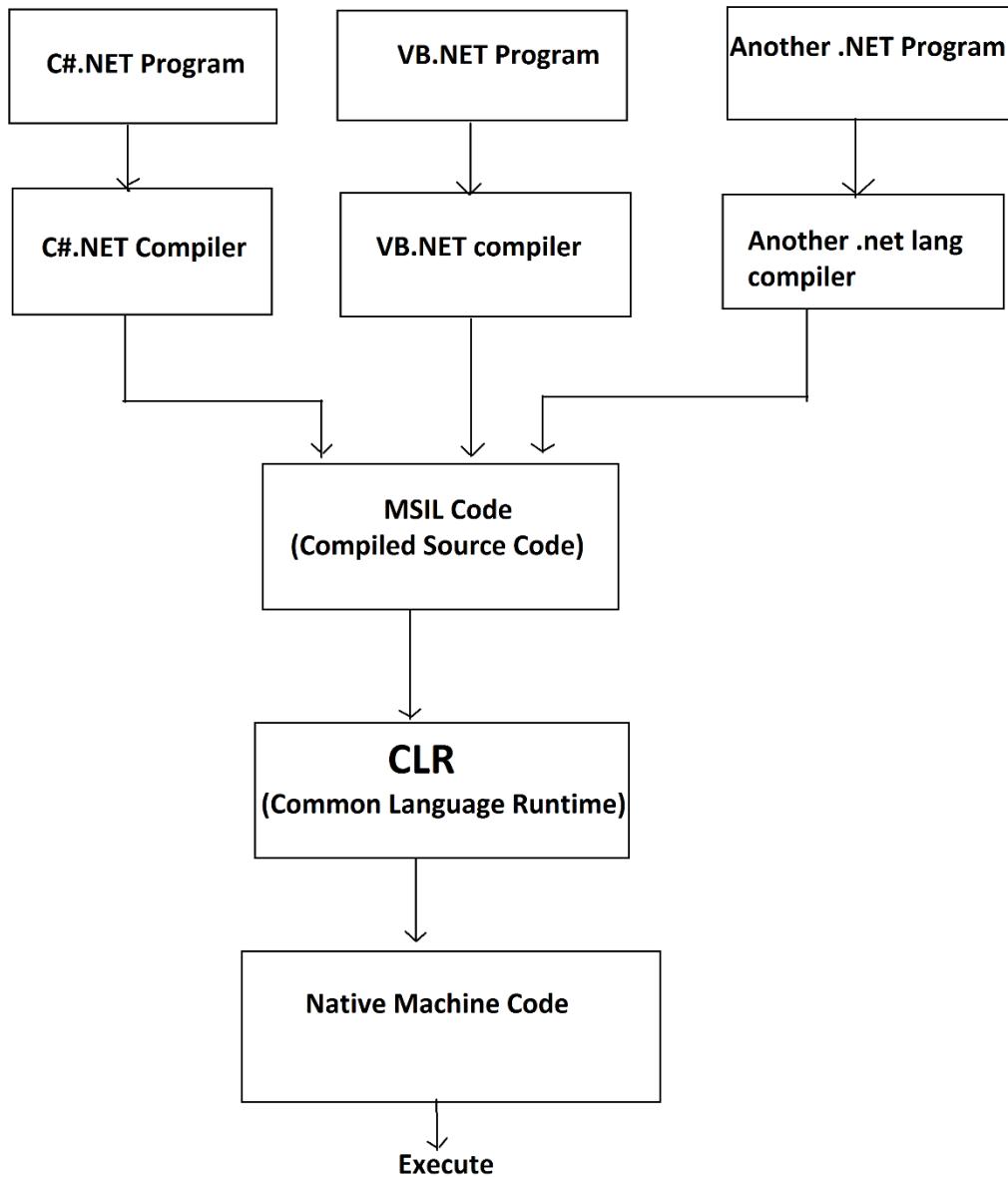
- 1998: Microsoft has completed developing the new language and named it as Simple Managed C (SMC).
- 1999: Microsoft renamed “SMS” as “C#” and wants to integrate C#, ASP. They named the integrated development platform as “NGWS” (Next Generation Windows Services).
- 2000: Microsoft renamed “NGWS” as “.NET” (Network-based). Microsoft announced “.NET” in July 2000.
- 2001: Microsoft has integrated C#.NET and ASP.NET in “.NET”.
- 2002: .NET 1.0
- 2003: .NET 1.1
- 2005: .NET 2.0
- 2006: .NET 3.0
- 2007: .NET 3.5
- 2010: .NET 4.0
- 2012: .NET 4.5
- 2013: .NET 4.5.1
- 2014: .NET 4.5.2
- 2015: .NET 4.6
- 2015: .NET 4.6.1
- 2016: .NET 4.6.2
- 2017: .NET 4.7
- 2017: .NET 4.7.1
- 2018: .NET 4.7.2
- 2019: .NET 4.8

Versions of .NET

Sl. No	.NET Framework	Year of release	Concepts
1	.NET Framework 1.0	2002	Console Apps, Language Fundamentals, OOP, WinForms, Web Forms, ADO.NET
2	.NET Framework 1.1	2003	ASP.NET Security
3	.NET Framework 2.0	2005	Improved web controls, Data controls, Themes and skins, Master pages, Partial classes, Nullable types, Anonymous methods, Generics
4	.NET Framework 3.0	2006	WPF WCF
5	.NET Framework 3.5	2007	LINQ ASP.NET AJAX
6	.NET Framework 4.0	2010	ADO.NET Entity Framework Task Parallel Library ASP.NET Web Pages ASP.NET MVC
7	.NET Framework 4.5	2012	Open Authentication ASP.NET Web API ASP.NET Bundles and Minification
8	.NET Framework 4.5.1	2013	Async and Await
9	.NET Framework 4.5.2	2014	Bug fixes
10	.NET Framework 4.6	2015	.NET Core ASP.NET Core
11	.NET Framework 4.6.1	2015	Spell Check for WPF
12	.NET Framework 4.6.2	2016	Soft keyboard support for WPF
13	.NET Framework 4.7	2017	Print API for WPF
14	.NET Framework 4.8	2019	Performance and Security Updates

CIL and CLR

Execution Model of .NET Programs



Steps:

1. Source Code: The source code of the program is written in ".cs" file.
2. Compilation Process: "CSC" (C Sharp Compiler) compiles (converts) the program from "C#.NET" (programmer understandable representation) to "MSIL (Microsoft Intermediate Language)" (also called as "byte code"). The "MSIL code" will be saved in ".exe" file. This

EXE file can't execute directly. The MSIL is neither understandable by the programmer, nor by operating system.

3. Execution Process: CLR (Common Language Runtime) converts the program from MSIL language at "EXE file" to "native machine language" (based on the current operating system). Operating System executes the "native machine language". Then we will get output.

CIL (Common Intermediate Language) or MSIL

- CIL or MSIL (Microsoft Intermediate Language) is the intermediate language, developed by Microsoft Corporation, for .NET.
- The .net programs are converted into "MSIL language" first; and then converted into "native machine language".
- The MSIL code will be stored in "EXE" file. Ex: filename.exe. The "native machine language" code will not be saved in any file, it directly runs.
- Once the code is converted into MSIL, it doesn't matter in which .net language it is originally developed. So for all .net languages, we can have a common runtime engine called "CLR".

CLR (Common Language Runtime)

- CLR stands for "Common Language Runtime".
- CLR is the "Execution Engine" or "Execution Environment" of .NET.
- To run any type of .net program (app), CLR must be installed in the computer.
- CLR will be started automatically when the .net application execution starts. CLR performs essential tasks internally, while running any .net application.
- Without CLR, we can't run any .net program.
- CLR reads the "MSIL code" from the EXE file, converts the same into "native machine code", gives the same to the operating system and then operating system executes the native

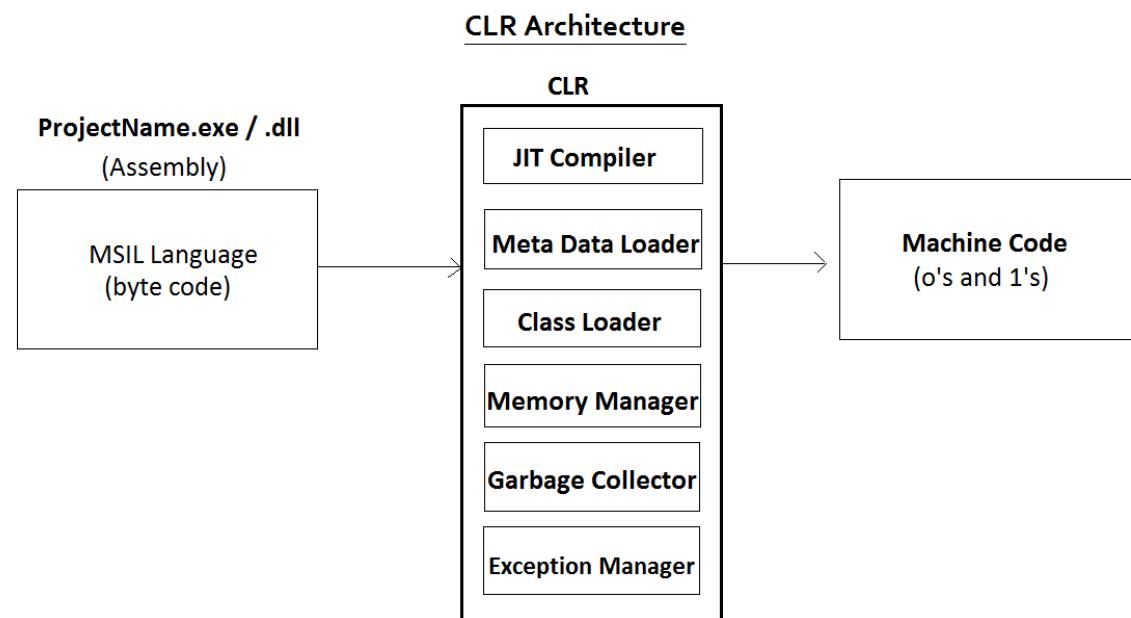
machine code; then the user gets the output. So CLR is helps the operating system while executing the .net program.

- CLR will be installed automatically as a part of “.NET Framework” software. CLR must be installed in both developer’s machine and user’s machine.

Sub Components of CLR

- CLR has the following inner components (sub components).
 1. **Memory Manager:** “Memory Manager” is a sub component in CLR, which allocates memory (in RAM) for the variables and objects in the program.
 2. **Garbage Collector:** “Garbage Collector” is a sub component in CLR, which deletes the variables and objects that are created during the program, automatically at the end of the program execution.
 3. **Class Loader:** “Class Loader” is a sub component in CLR, which loads a class on-demand. When we try to access a class in the program for the first time, then the “Class Loader” searches for the class in the entire program, loads the class into the memory, and it passes the class to JIT compiler. That means if we don’t call a class, it will not be loaded into memory. This avoids un-necessary loading of the classes, if we don’t require them. So it improves performance.
 4. **JIT (Just-In-Time) Compiler:** “JIT Compiler” is a sub component in CLR, which converts the “MSIL code” into “native machine language” (based on current operating system).
 5. **Thread Manager:** “Thread Manager” is a sub component in CLR, which manages the threads of the program. It gives necessary instructions to the processor, which thread is to be executed when. A thread is a "part of the program" or "background work".

6. Exception Manager: “Exception Manager” is a sub component in CLR, which passes necessary instructions to the operating system, which code should be executed when an exception (runtime error) occurs while executing the program.
7. Security Manager: “Security Manager” is a sub component in CLR, which takes care about different types of security in .net such as windows authentication, forms authentication, open authentication etc.



.NET Framework Architecture

What is .NET Framework

- It is the “Software Development Kit (SDK)”, which contains many components such as CIL, CLR, FCL, CLS, which is used to develop & run the .net applications.
- It should be installed in both developer system and user system.
- .NET Framework download link:

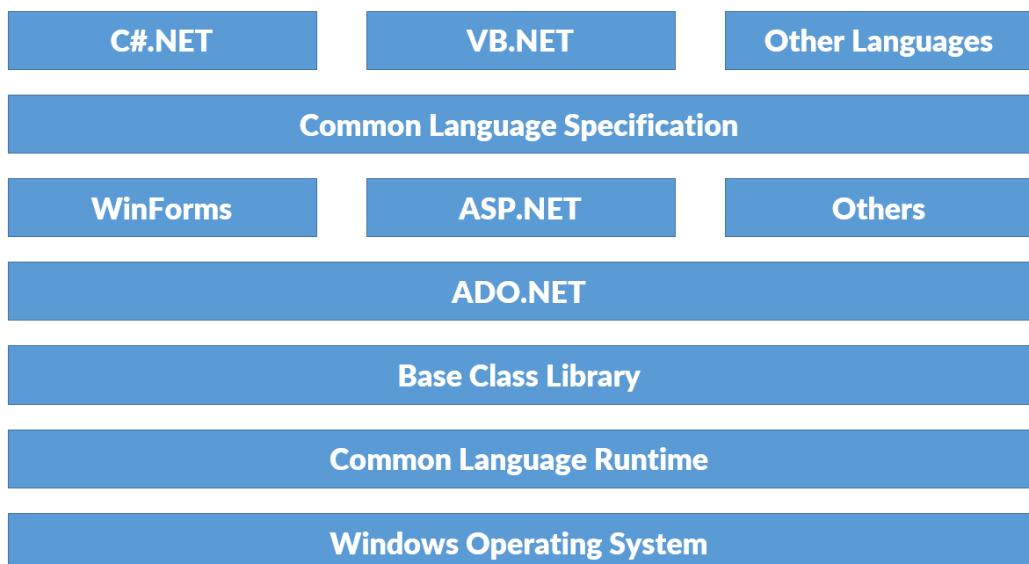
<https://www.microsoft.com/en-us/download/details.aspx?id=55167>

System Requirements of .NET Framework:

Software / Hardware Requirement	Minimum	Recommended
Processor	1.6 Ghz	2.3 Ghz or higher
RAM	1 GB	2 GB or higher
Hard disk free space on C: drive	2 GB	4 GB or higher
Operating System	Windows 7 + Service Pack 1 Windows 8 Windows 8.1 Windows 10	Windows 10

.NET Framework Architecture

- “.NET framework” is divided as several components.
- “.NET Framework Architecture” explains the list of components of .NET Framework, and how they are arranged internally.



Explanation:

- “Windows” operating system works based on “processor”.
- “CLR” works based on “Windows Operating System”.
- “BCL” works based on “CLR”.
- “ADO.NET” works based on “BCL”.
- “WinForms”, “ASP.NET”, and “Other Frameworks” work based on “ADO.NET” and “BCL”.
- “CLS” is created based on “WinForms”, “ASP.NET” and “Other Frameworks”.
- “C#.NET”, “VB.NET”, “Other Languages” are developed based on “CLS”.
- Finally, the programmers are writing the programs by using the languages called “C#.NET”, “VB.NET” etc.

CLS and CTS

CLS (Common Language Specification)

- “CLS” is the set of rules, based on which all .net languages (C#.NET, VB.NET, VC++.NET etc.) are developed.
- The common rules are about literals, operators, identifiers, data types, type conversion, object oriented programming etc.

CTS (Common Type System)

- “CTS” is a set common data types, based on which, the data types of all .net languages (C#.NET, VB.NET, VC++.NET etc.) are developed.
- Thus, we achieve the uniform data types among all .net languages.
- The following is the list of data types of CTS:
 1. SByte
 2. Byte
 3. Short
 4. UShort
 5. Int32
 6. UInt32
 7. Int64
 8. UInt64
 9. Single
 10. Double
 11. Decimal
 12. Char
 13. String
 14. Boolean

FCL and BCL

FCL (Framework Class Library)

- .NET provides a set of classes and interfaces, based on which we can develop .net applications.
- The “DLL (Dynamic Link Library)” file is a collection of namespaces; Namespace is a collection of classes and interfaces.
- FCL is divided into the following parts:
 1. BCL: BCL is a set of classes and interfaces, which can be used in all types of applications.
 2. WinForms: This is a set of classes and interfaces, which can be used only in windows applications.
 3. ASP.NET: This is a set of classes and interfaces, which can be used only in web applications.
 4. ADO.NET: This is a set of classes and interfaces, which can be used in all types of applications for connecting to databases.

Introduction to Visual Studio

What is Visual Studio

- Visual Studio is the “IDE” (Integrated Development Environment), where you can write all types of .net programs (C#.NET, ASP.NET).
- It provides advantages such as syntax highlighting, intellisense, deployment etc.

Versions of Visual Studio

Visual Studio Version	Year of Release	Supported Operating Systems	Supported .NET Framework Versions
Visual Studio 2002 (7.0)	2002	Windows 2000 Windows XP	.NET 1.0
Visual Studio 2003 (7.1)	2003	Windows 2000 Windows XP	.NET 1.1
Visual Studio 2005 (8.0)	2005	Windows 2000 Windows XP	.NET 2.0
Visual Studio 2008 (9.0)	2007	Windows 2000 Windows XP Windows 7	.NET 2.0 .NET 3.0 .NET 3.5
Visual Studio 2010 (10.0)	2010	Windows XP + Service Pack 3 Windows 7	.NET 2.0 .NET 3.0 .NET 3.5 .NET 4.0
Visual Studio 2012 (11.0)	2012	Windows 7 Windows 8.1 Windows 10	.NET 2.0 .NET 3.0 .NET 3.5 .NET 4.0 .NET 4.5
Visual Studio 2013 (12.0)	2013	Windows 7 + Service Pack 1 Windows 8.1	.NET 2.0 .NET 3.0

C#.NET 8.0

		Windows 10	.NET 3.5 .NET 4.0 .NET 4.5 .NET 4.5.1
Visual Studio 2015 (14.0)	2015	Windows 7 + Service Pack 1 Windows 8.1 Windows 10	.NET 2.0 .NET 3.0 .NET 3.5 .NET 4.0 .NET 4.5 .NET 4.5.1 .NET 4.5.2 .NET 4.6 .NET 4.6.1 .NET 4.6.2
Visual Studio 2017 (15.0)	2017	Windows 7 + Service Pack 1 Windows 8.1 Windows 10	.NET 2.0 .NET 3.0 .NET 3.5 .NET 4.0 .NET 4.5 .NET 4.5.1 .NET 4.5.2 .NET 4.6 .NET 4.6.1 .NET 4.6.2 .NET 4.7
Visual Studio 2019			

System Requirements of Visual Studio 2019:

Software / Hardware Requirement	Minimum	Recommended
Processor	1.6 Ghz	2.3 Ghz or higher
RAM	2 GB	4 GB or higher
Hard disk free space on	10 GB	15 GB or higher

C: drive		
Operating System	Windows 7 + Service Pack 1 Windows 8.1 Windows 10	Windows 10
Internet Explorer	Internet Explorer 10	Internet Explorer 11 or above

Project

- Project is a folder, which is a “group of files”.
- A file contains code (program).
- When we compile the project, Visual Studio generates only one EXE file for the entire project, which contains the compiled source code (in MSIL language) of all the files of the same project.

Solution

- Solution is a folder, which is a “group of projects”.

File Types of .NET

- .NET supports the following file types:

File Extension	Full form	Description
.cs	C# file	A C# file contains the C#.NET program (source code).
.vb	VB file	A VB file contains the VB.NET program (source code).
.csproj	C# Project	A project is a collection of files. The project file contains the list of all the files in the current project.
.sln	Solution	A solution is a collection of projects. The solution file contains the list of all the projects in the current solution.

.exe	Executable file	The EXE file contains the compiled source code of a project. For every project, a separate EXE file will be created.
.config	Configuration File	The configuration file contains the configuration settings of a project.

Folder Structure of .NET Project in Visual Studio

- .NET program (project) should have the following folder structure.
- Every solution is a folder. Every project is folder.

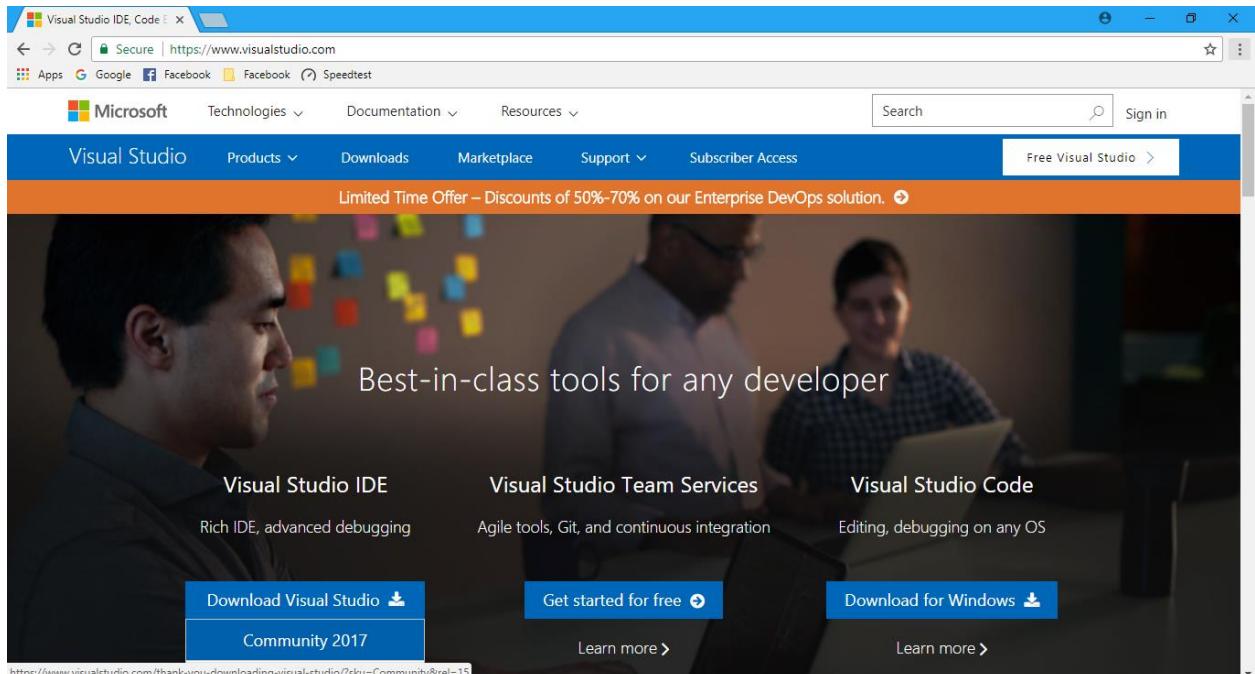


Visual Studio - Installation

Steps to install “Visual Studio 2019”:

Go to <http://www.visualstudio.com>

Click on “Download Visual Studio” – “Community 2019”.

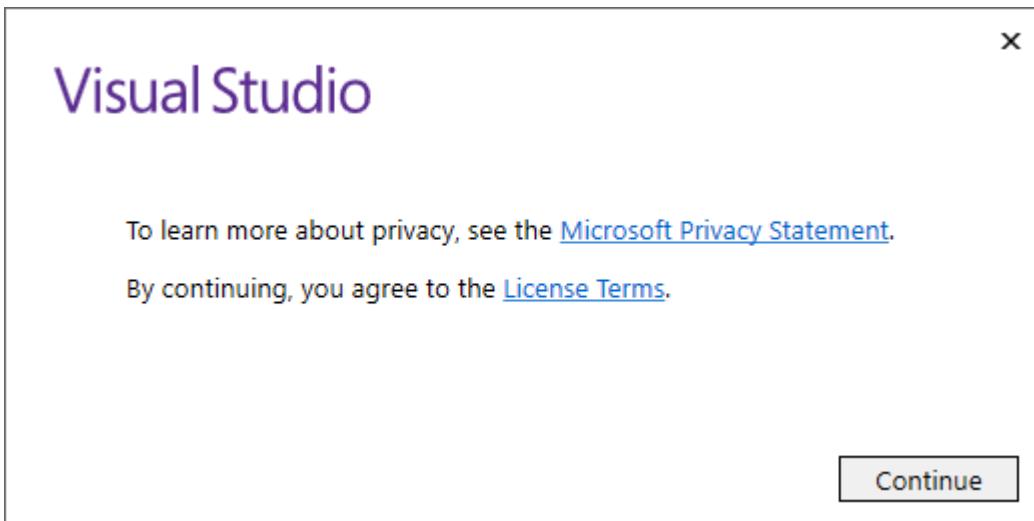


You will get “vs_community.exe” file.

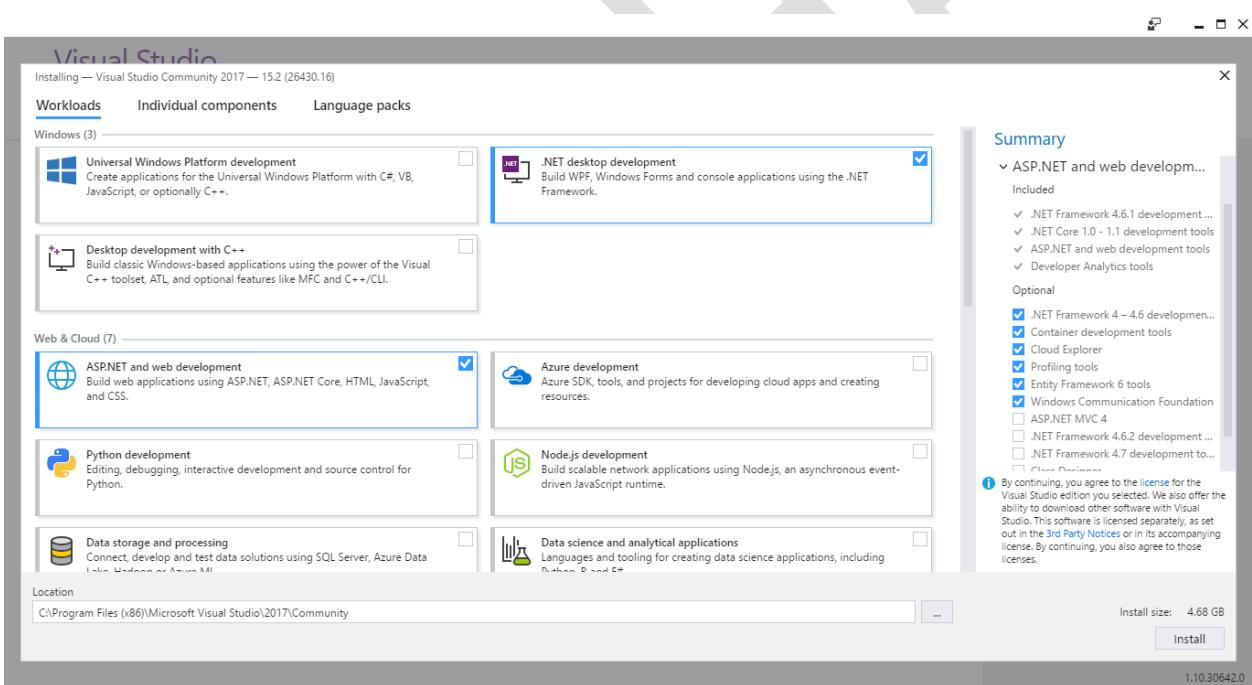
Run “vs_community.exe” file.

Click on “Run”.

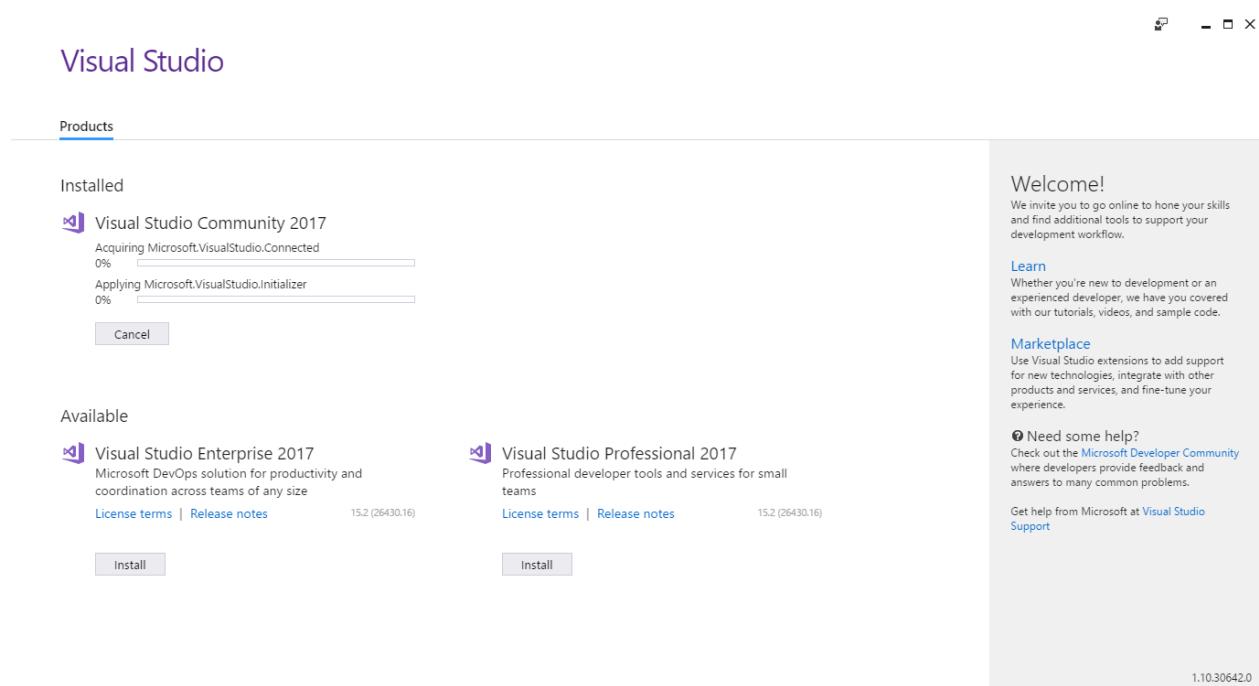
Click on “Continue”.



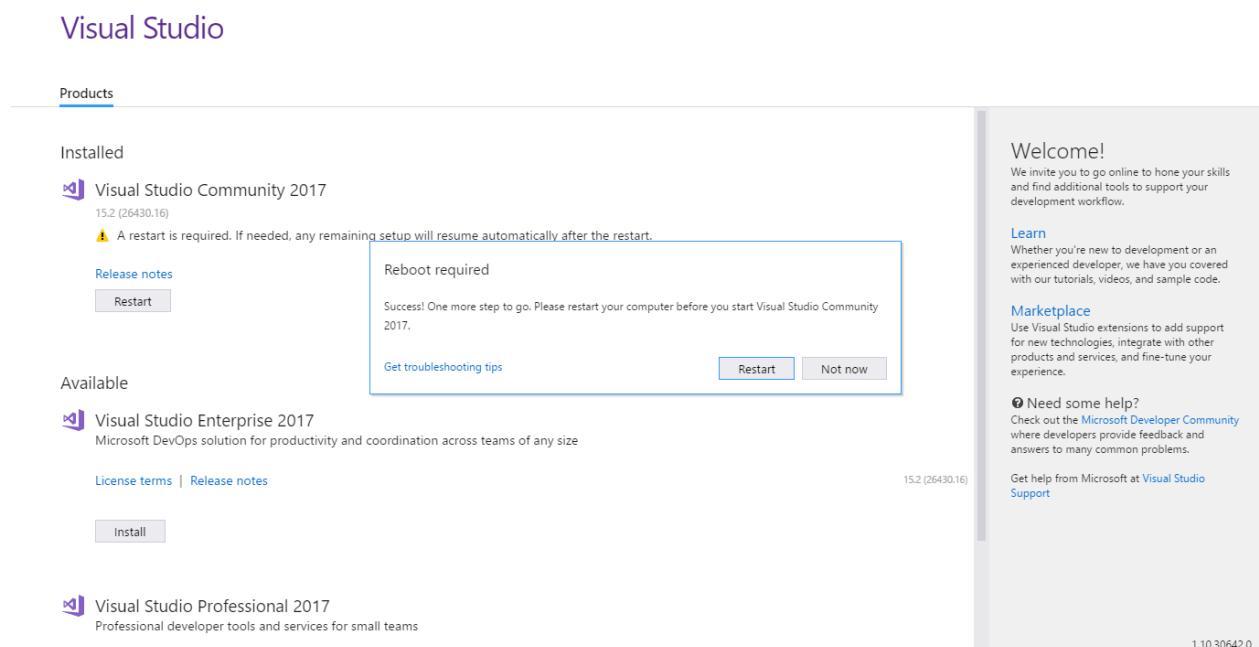
Check the checkboxes “.NET desktop development” and “ASP.NET and web development”.



Click on “Install”.

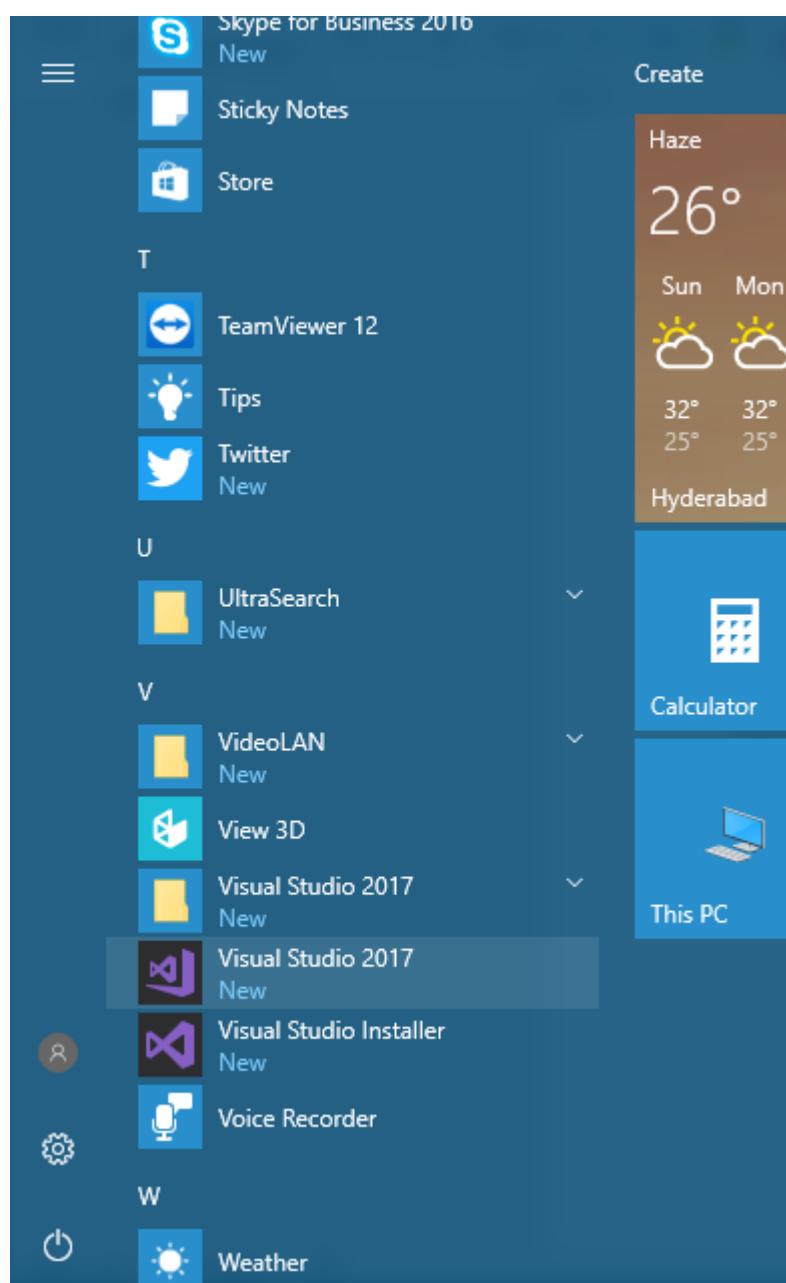


Wait for installation to be completed.



Click on “Restart”.

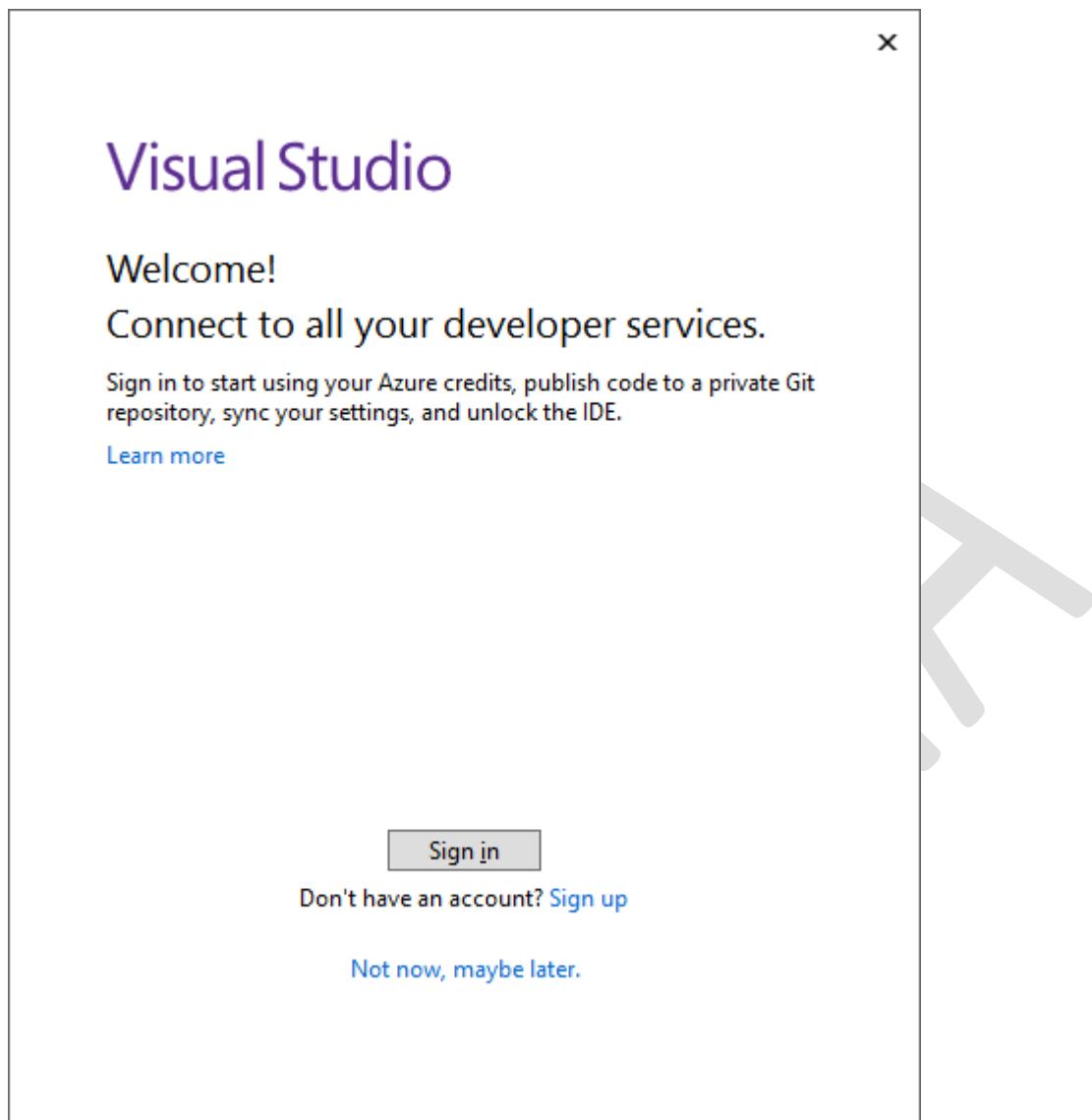
After restarting your system, go to “Start” > “Visual Studio 2019”.



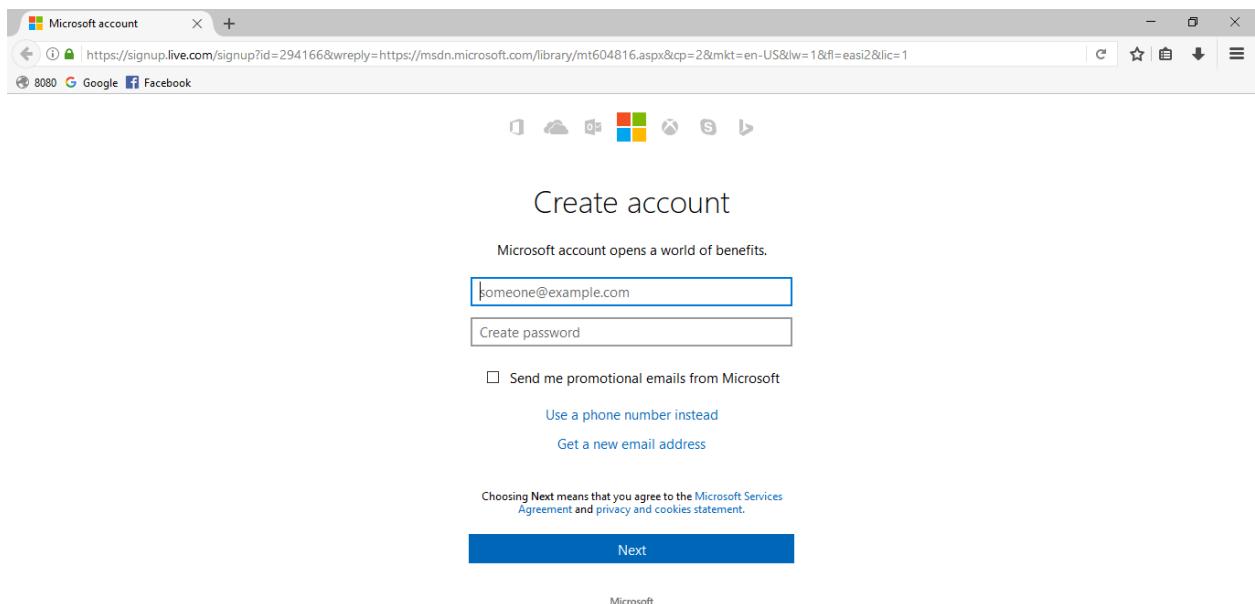
If you have Microsoft account already and click on “Sign in” and complete the login process.

If you don't have Microsoft account, click on “Sign up” and complete the registration process.

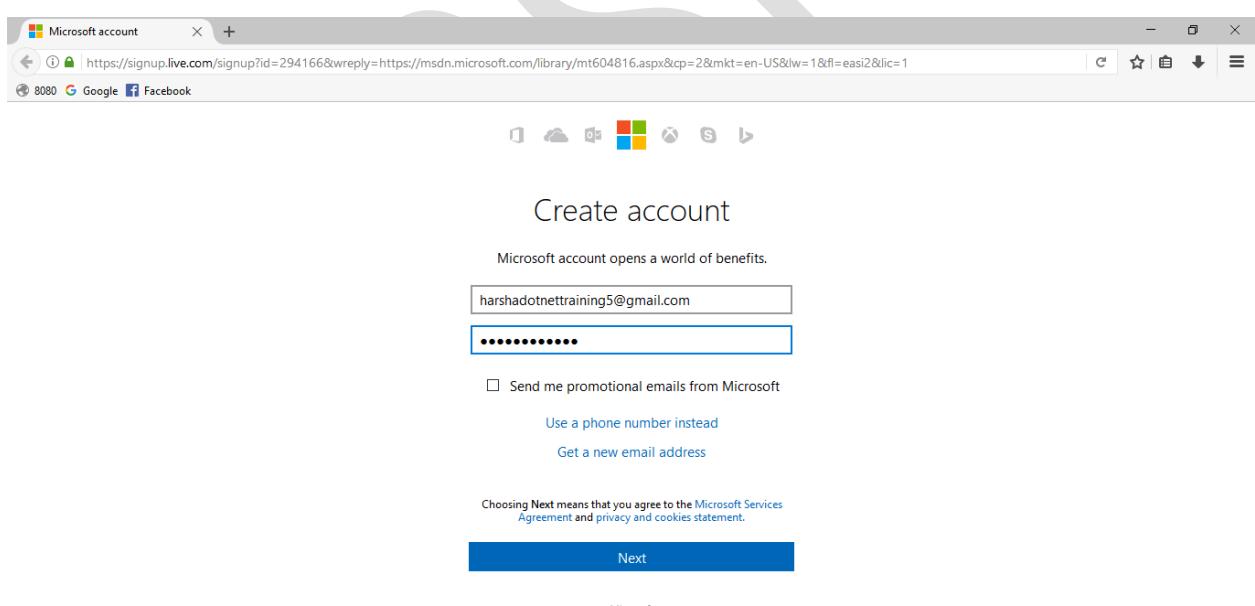
Note: If you don't want to login, click on “Not now, may be later”; but then Visual Studio expires in 30 days.



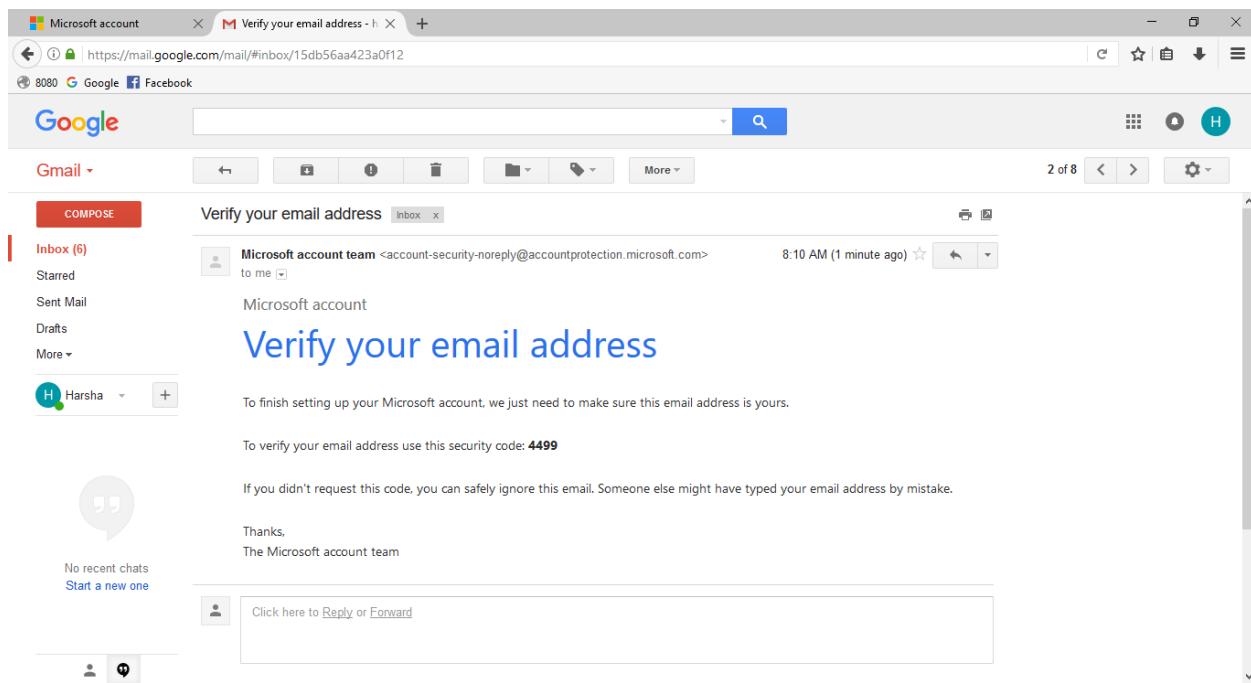
If you click on "Sign up", you will get the following page.



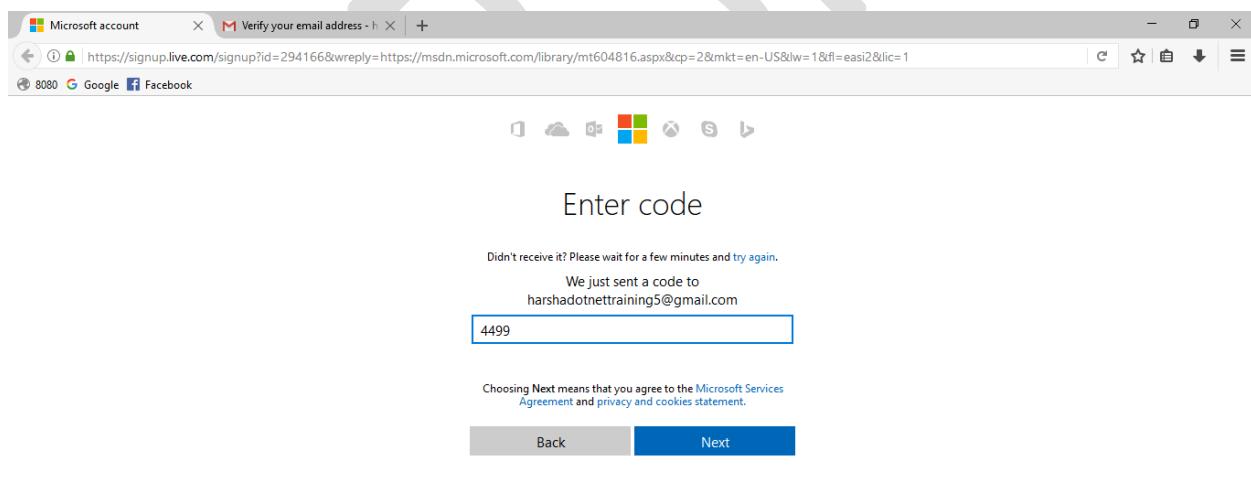
Enter your email id and enter new password.



Click on “Next”.



You will get a security code to your email. Login into your gmail, check the code that you have received and enter it in this page.



After entering the security code, click on “Next”.

The screenshot shows a browser window with the URL <https://msdn.microsoft.com/library/mt604816.aspx>. The page content reads:

Return to Visual Studio to sign in to the IDE with your new Microsoft account!

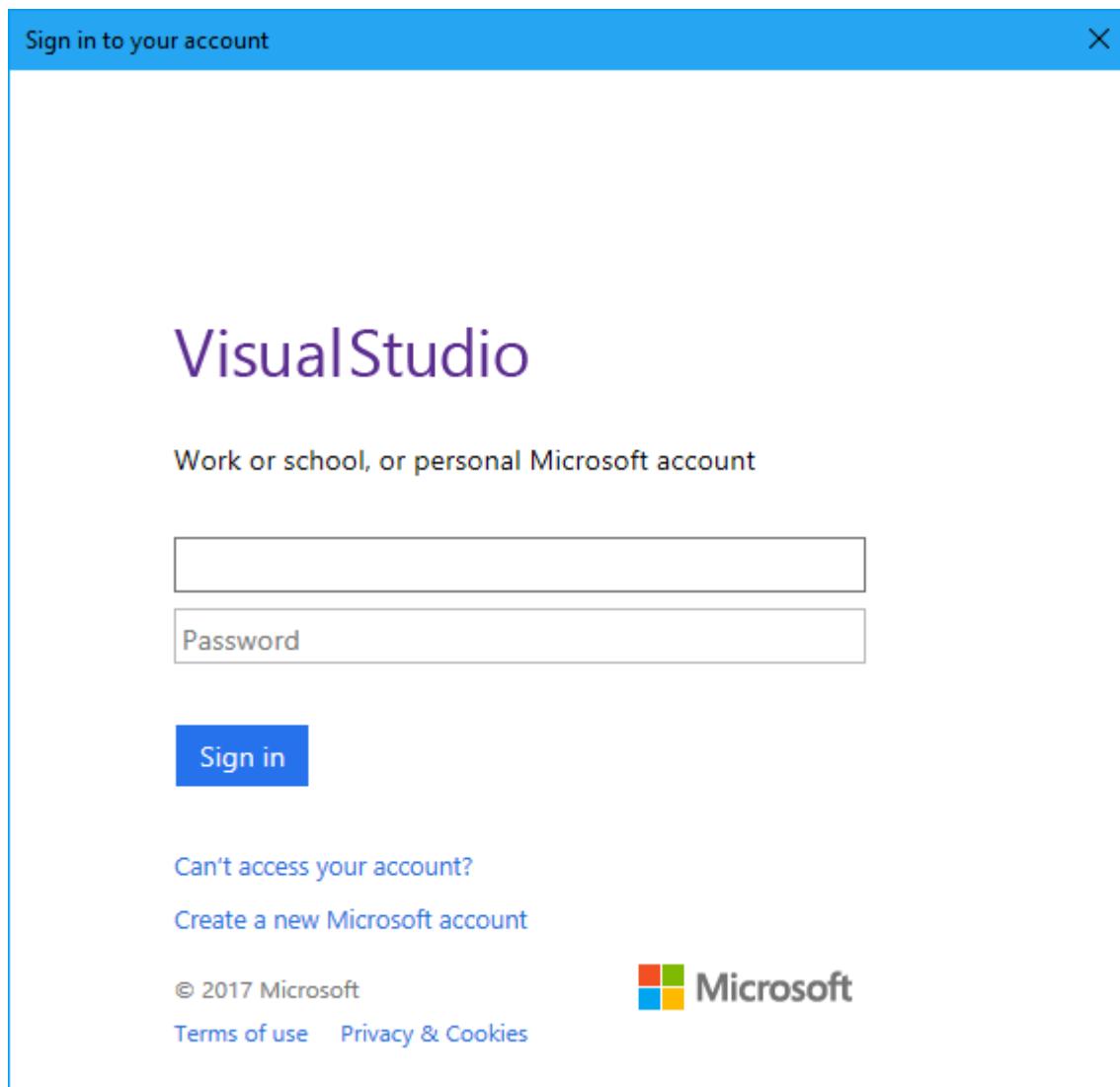
You have successfully created a Microsoft Account. Return to Visual Studio and sign in from the Welcome wizard during first launch, or from the upper right corner of the IDE any time.

Sign in to the IDE to start using your Azure credits, publish code to a private Git repository, sync your settings, and unlock the IDE. [Learn more](#)

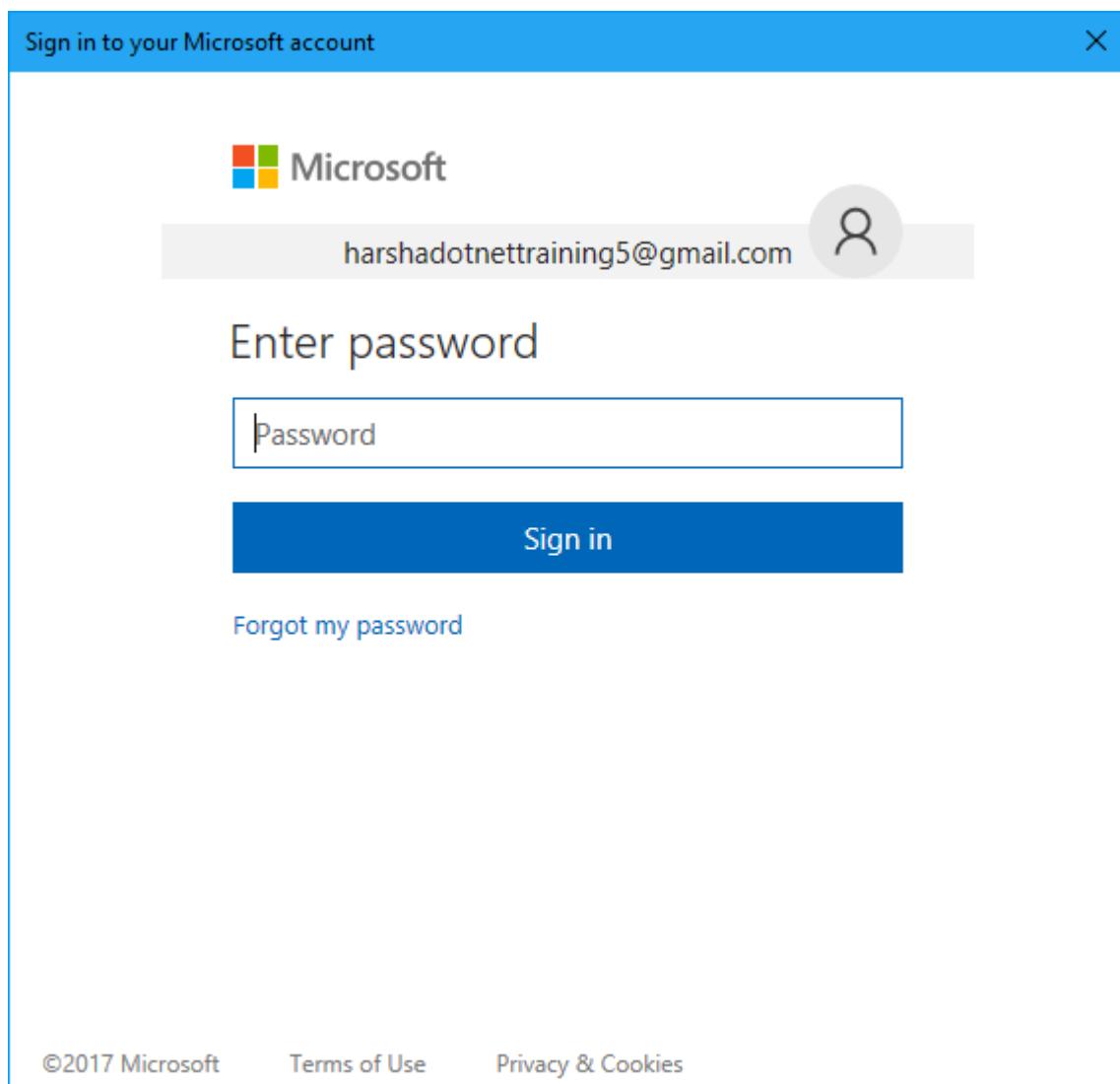
A screenshot of the Visual Studio IDE interface is shown, with the "Sign in" button in the top right corner circled in red.

If you get the above page, you have successfully registered.

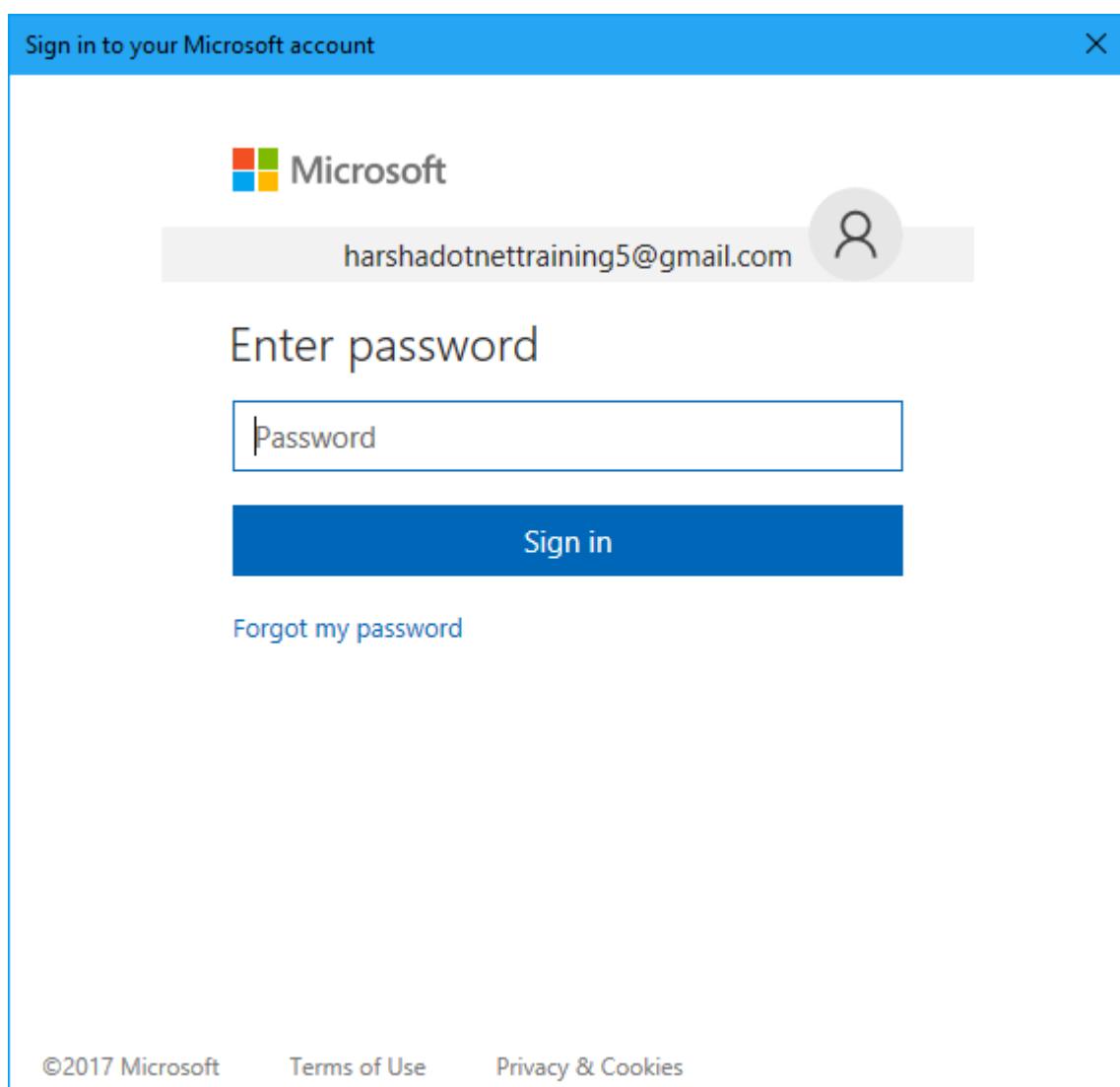
Now click on “Sign in” in Visual Studio and enter the email.

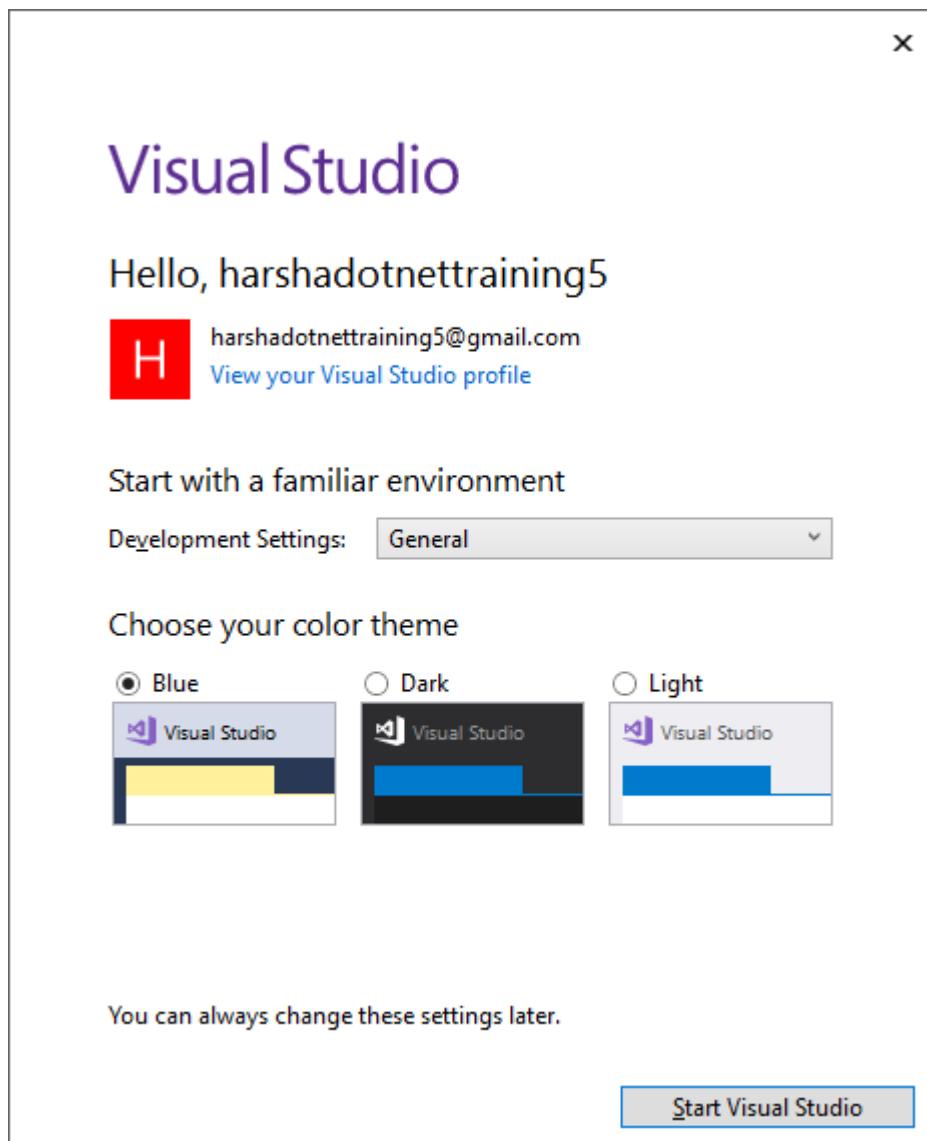


Enter the password.



Click on "Sign in".



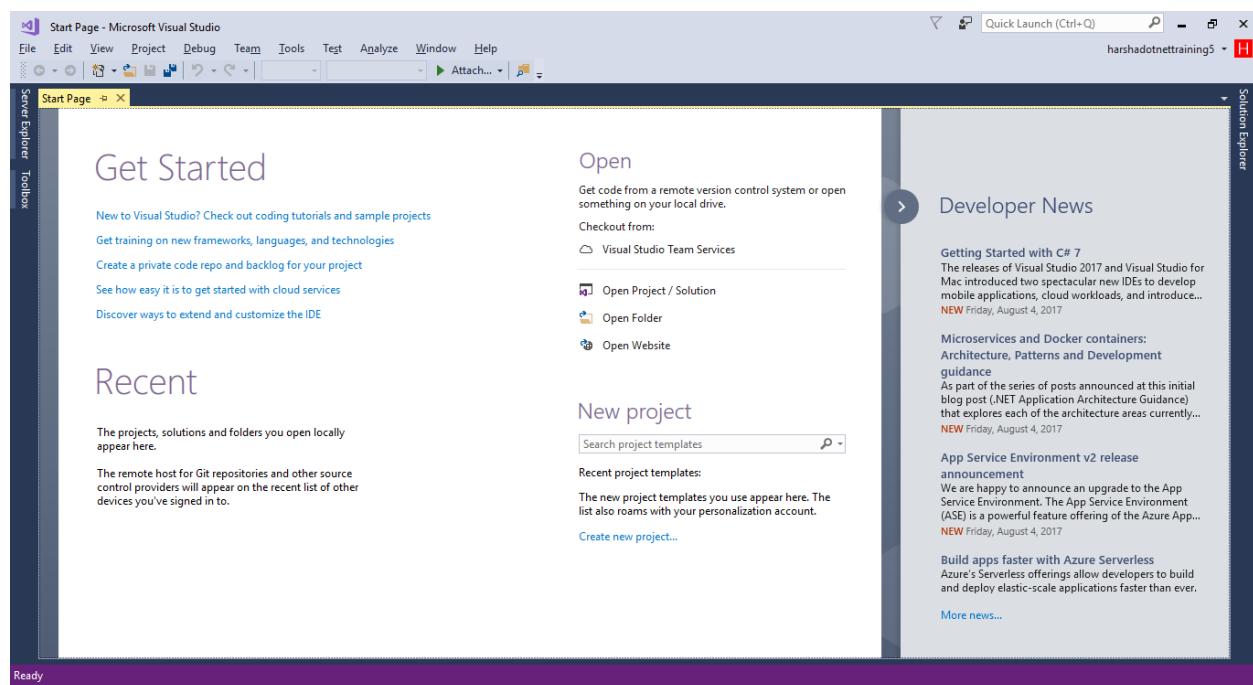


Select “Development Settings” as “General”.

Select “color theme” as “Blue”.

Click on “Start Visual Studio”.

C#.NET 8.0



Visual Studio Installation has been completed successfully.

C#.NET – Language Fundamentals

Introduction to C#.NET

What is C#.NET:

C#.NET is the .NET's most popular programming language, which is used to create stand-alone applications (console applications and windows applications, windows services) primarily.

- C#.NET is developed on basis of C, C++.
- C#.NET is an “Object Oriented Programming Language”.
- C#.NET is a case sensitive language.
- C#.NET is an advanced and matured language.
- C#.NET is a high-level programming language.
- C#.NET is a compiler-based language.
- C#.NET is a part of .NET Framework.
- C#.NET programs run based on the CLR (.NET run time environment).

Versions of C#.NET

Sl. No	C#.NET Version	.NET Framework Version	Year of release
1	C#.NET 1.0	.NET Framework 1.0	2002
2	C#.NET 1.1	.NET Framework 1.1	2003
3	C#.NET 2.0	.NET Framework 2.0	2005
4	C#.NET 3.0	.NET Framework 3.0	2006
5	C#.NET 3.5	.NET Framework 3.5	2007
6	C#.NET 4.0	.NET Framework 4.0	2010
7	C#.NET 5.0	.NET Framework 4.5	2012
8	C#.NET 5.1	.NET Framework 4.5.1	2013
9	C#.NET 5.2	.NET Framework 4.5.2	2014
10	C#.NET 6.0	.NET Framework 4.6	2015
11	C#.NET 7.0	.NET Framework 4.7	2017
12	C#.NET 8.0	.NET Framework 4.8	2019

Tokens of C#.NET

- In "C#.NET", we have to use the following tokens (parts of the language).

Sl. No	Token	Description
1	Keywords	abstract, as, base, bool, break, byte, case, catch, char, class, const, continue, decimal, default, delegate, do, double, else, enum, event, false, finally, float, for, foreach, goto, if, in, int, interface, internal, is, long, namespace, new, null, object, out, override, private, protected, public, readonly, ref, return, sbyte, sealed, short, sizeof, static, string, struct, switch, this, throw, true, try, typeof, uint, ulong, ushort, using, virtual, void, while, async, await, from, join, let, orderby, partial, set, get, value, var, where
2	Operators	Any symbols such as +, -, *, /, %, =, == etc.

3	Literals	<p>Fixed values are literals.</p> <ul style="list-style-type: none"> • Integer literals: Any number without decimal part. Ex: 10 • Floating-point literals: Any number with decimal part. Ex: 10.87 • Character literals: Any single character in single quotes. Ex: 'A'. The character can be alphabet, digit, space or special symbol. • String literals: One or more characters in double quotes. Ex: "Abc 123\$" • Boolean literals: true / false
4	Identifiers	Any user-defined names: abc

Naming Conventions for Identifiers

- Class names, Interface names, Namespace names, Structure names, Method names, Property names: PascalCasing
- Local variables: camelCasing
- Public Fields: PascalCasing
- Private Fields: camelCasing with underscore (_)

Identifier Naming Rules

You must follow the below rules while giving name for identifier.

- Identifier names can't be same as keyword.
- Identifier names should not have spaces.
- Identifier names should not have special characters.
- Duplicate identifier names not allowed.

Console Applications - Introduction

What is Console Application?

- Console applications are the programs that run on “Command Prompt” window.
- Console applications support CUI (Character User Interface).
- In console applications, all the input and output will be in the form of characters only.
- Console applications are not user-friendly.
- Console applications are not used in real time, but good for learning programming basics and OOP concepts.

Rules of Console Application

- Console application must have atleast one class; which is called as “Main class”.
- The “Main class” must contain atleast one method; which is called as “Main method”.
- The “main method” name should be “Main”; “M” is capital letter.
- When the program execution starts the “Main” method will be automatically executed.
Main method is the “starting point of the program execution”.
- The Main method should be “static” method, because it should be called without creating any object for the class.

Syntax:

```
class classname
{
    static void Main()
    {
    }
}
```

Console Application - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “Project1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “Solution1”.
- Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

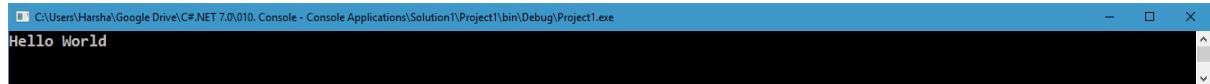
Program.cs

```
class Sample
{
    static void Main()
    {
        System.Console.WriteLine("Hello World");
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The “System.Console” class

The “System.Console” class

- C#.NET provides “System.Console” class to perform I/O operations in console applications.
- “System” is a namespace; “Console” is a class.
- The following are the important methods of “System.Console” class.
 1. System.Console.WriteLine()
 2. System.Console.ReadLine()
 3. System.Console.ReadKey()
 4. System.Console.ReadKey(true)
 5. System.Console.Clear()

1. System.Console.WriteLine()

- This statement is used to display the given value on the command prompt window.
- After printing the value, the cursor will be kept in the same line.
- Syntax: System.Console.WriteLine(*value*);
- Example: System.Console.WriteLine(100);

2. System.Console.WriteLine()

- This statement is used to display the given value on the command prompt window.
- After printing the value, the cursor will be automatically moved to the next line.
- Syntax: `System.Console.WriteLine(value);`
- Example: `System.Console.WriteLine(100);`

3. System.Console.ReadKey()

- This statement is used to wait until the user presses any key on the keyboard.
- Syntax: `System.Console.ReadKey();`
- Example: `System.Console.ReadKey();`

4. System.Console.ReadLine()

- This statement is used to accept a string value from keyboard.
- Syntax: `System.Console.ReadLine();`
- Example: `System.Console.ReadLine();`

5. System.Console.Clear()

- This statement is used to clear the screen.
- Syntax: `System.Console.Clear();`
- Example: `System.Console.Clear();`

System.Console.WriteLine – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “WriteLineExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WriteLineExample”.
- Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

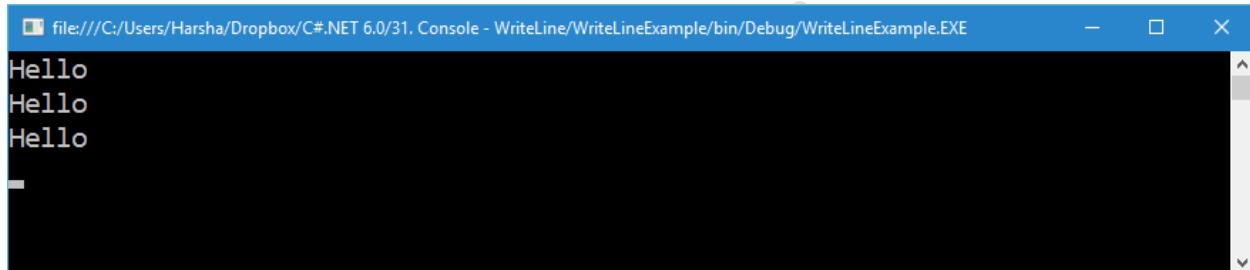
Program.cs

```
class Program
{
    static void Main()
    {
        System.Console.WriteLine("Hello");
        System.Console.WriteLine("Hello");
        System.Console.WriteLine("Hello");
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



System.Console.ReadKey – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReadKeyExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ReadKeyExample”.
- Click on OK.
- It shows “Program.cs” file automatically.

- Type the following code for “Program.cs” file:

Program.cs

```
class Program
{
    static void Main()
    {
        //display message
        System.Console.WriteLine("Hello");

        //wait for pressing any key on the keyboard
        System.Console.ReadKey();

        //display message
        System.Console.WriteLine("Hello");

        //wait for pressing any key on the keyboard
        System.Console.ReadKey();

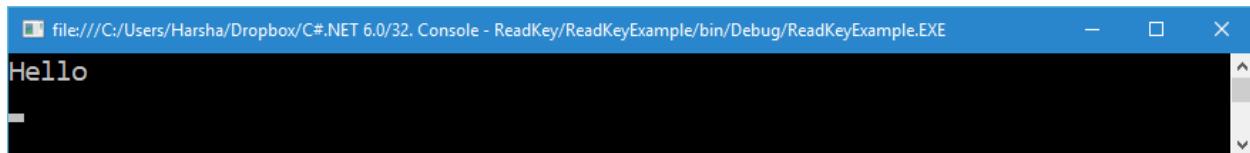
        //display message
        System.Console.WriteLine("Hello");

        //wait for pressing any key on the keyboard
        System.Console.ReadKey();
    }
}
```

Running the Project

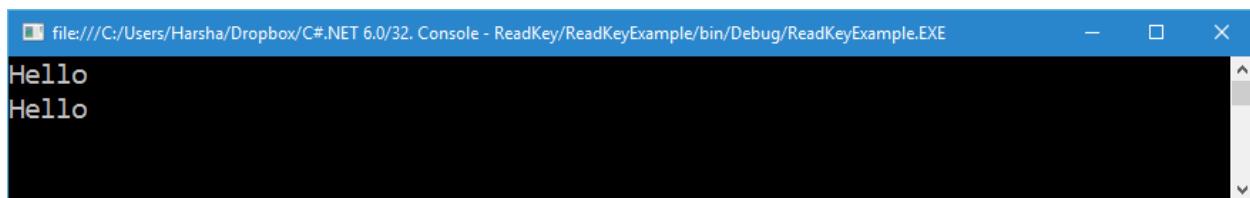
- Go to “Debug” menu and click on “Start Debugging”.

Output



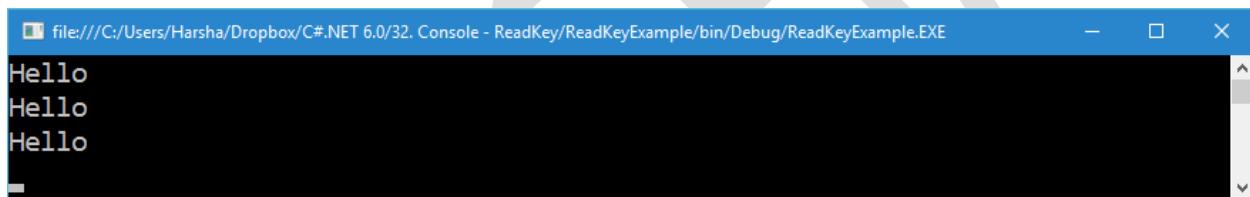
```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/32. Console - ReadKey/.ReadKeyExample/bin/Debug/ReadKeyExample.EXE
Hello
-
```

Press Enter.



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/32. Console - ReadKey/.ReadKeyExample/bin/Debug/ReadKeyExample.EXE
Hello
Hello
-
```

Press Enter.



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/32. Console - ReadKey/.ReadKeyExample/bin/Debug/ReadKeyExample.EXE
Hello
Hello
Hello
-
```

System.Console.Clear - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ClearExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “ClearExample”.
- Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

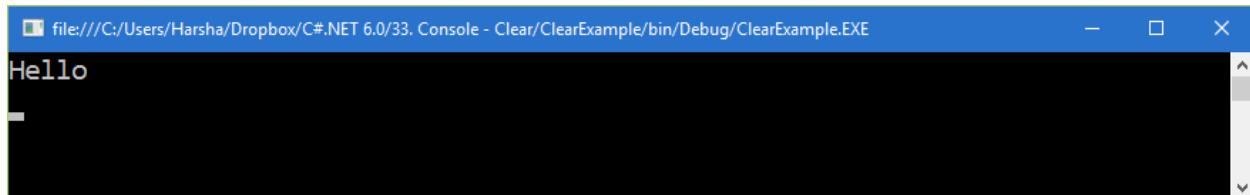
Program.cs

```
class Program
{
    static void Main()
    {
        //display message
        System.Console.WriteLine("Hello");
        //wait for pressing any key on the keyboard
        System.Console.ReadKey();
        //clear the screen
        System.Console.Clear();
        //display message
        System.Console.WriteLine("how");
        //wait for pressing any key on the keyboard
        System.Console.ReadKey();
        //clear the screen
        System.Console.Clear();
        //display message
        System.Console.WriteLine("are you");
        //wait for pressing any key on the keyboard
        System.Console.ReadKey();
    }
}
```

Running the Project

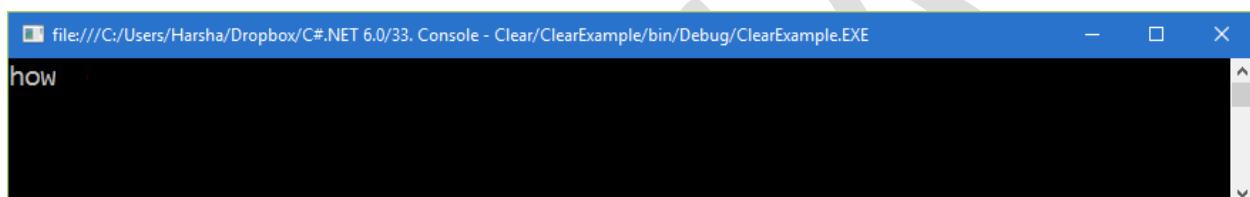
- Go to “Debug” menu and click on “Start Debugging”.

Output



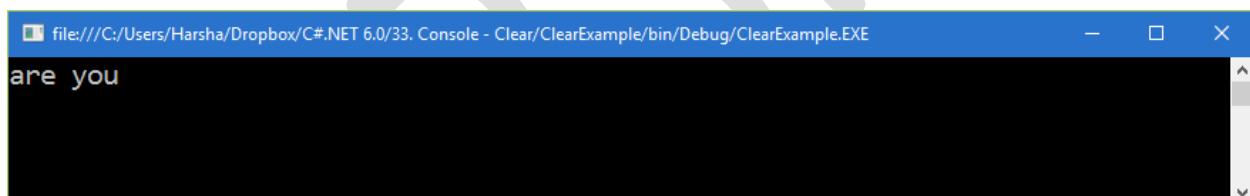
```
Hello
```

Press Enter.



```
how
```

Press Enter.



```
are you
```

Variables

Variables

- A variable is a named memory location in RAM, to store a particular type of value temporarily while the program is running.
- All the variables stored in RAM (temporary memory).
- All the variables must be declared before its usage. While declaring variables, data type is to be specified. Based on the data type, the amount of memory to be allocated will be decided. Once a variable is declared, we can't change the "variable name" or "variable's data type".
- The variables memory will be allocated when the program execution starts; and all the variables will be deleted (de-allocated) from memory automatically, at the end of the program.
- A variable can store only one value. If you assign another value, the old value will be overwritten.
- We can change the value of a variable any no. of times.
- Syntax to create a variable:

datatype variablename;

- Syntax to set value into a variable:

variablename = value;

- Syntax to get the value of a variable:

variablename

Variables - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “VariablesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “VariablesExample”.
- Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

Program.cs

```
class Program
{
    static void Main()
    {
        //create a variable
        int x;
        //set value into the variable
        x = 10;
        //get the value of variable
        System.Console.WriteLine(x);
        //wait for pressing any key on the keyboard
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/34. Console - Variables/VariablesExample/bin/Debug/VariablesExample.EXE
10
```

Data Types

Data Type

- A data type is a concept, which specifies the type of the data that is to be stored in a variable.

Types of Data types

- Data types are two types:
 - I. Numerical data types
 - II. Non-numerical data types

Numerical Data Types

Sl. No	Data Type	Description	No. of Bytes	Range	Suitable to store
1	sbyte	8-bit signed integer	1 byte	-128 to 127	Very small positive or negative integers
2	byte	8-bit unsigned integer	1 byte	0 to 255	Very small positive integers
3	short	16-bit signed integer	2 bytes	-32,768 to 32,767	Small positive or negative integers
4	ushort	16-bit unsigned integer	2 bytes	0 to 65,535	Small positive integers

5	int	32-bit signed integer	4 bytes	-2,147,483,648 to 2,147,483,647	Medium positive or negative integers
6	uint	32-bit unsigned integer	4 bytes	0 to 4,294,967,295	Medium positive integers
7	long	64-bit signed integer	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,770,000	Large positive or negative integers
8	ulong	64-bit unsigned integer	8 bytes	0 to 18,446,744,073,709,551,615	Large positive integers
9	float	Signed floating-point number	4 bytes	-3.402823E+38 to 3.402823E+38 (Precision: 7 digits)	Small floating-point numbers
10	double	Signed floating-point number	8 bytes	-1.79769313486232E+308 to 1.79769313486232E+308 (Precision: 15 digits)	Medium floating-point numbers
11	decimal	Signed floating-point number	16 bytes	-79228162514264337593543950335 to 79228162514264337593543950335 (Precision: 28 digits)	Large floating-point numbers

Default Numerical data types

- C# compiler automatically treats a “number without decimal part” as “int” data type, if it is within the maximum limit of “int” data type.

- C# compiler automatically treats a “number without decimal part” as “long” data type, if it exceeds the limit of “int” data type.
- C# compiler automatically treats a number with decimal part as “double” data type.

Non-Numerical Data Types

Sl. No	Data Type	Description	Value format	No. of Bytes
1	char	To store single character	'character'	2 bytes
2	string	To store one or more characters. <u>Max:</u> 2 billion characters	"string here"	No. of characters * 2
3	bool	To store true / false values.	true or false	1 bit

Numerical Data Types - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “NumericalDataTypesExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “NumericalDataTypesExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create variables for all numerical data types
        sbyte a = 10;
        byte b = 20;
        short c = 30;
        ushort d = 40;
        int e = 50;
        uint f = 60;
        long g = 70;
        ulong h = 80;
        float i = 90.23F;
        double j = 100.23489;
        decimal k = 110.882932M;

        //displays the values of all variables
        System.Console.WriteLine(a);
        System.Console.WriteLine(b);
        System.Console.WriteLine(c);
        System.Console.WriteLine(d);
        System.Console.WriteLine(e);
        System.Console.WriteLine(f);
        System.Console.WriteLine(g);
        System.Console.WriteLine(h);
        System.Console.WriteLine(i);
        System.Console.WriteLine(j);
        System.Console.WriteLine(k);
        System.Console.ReadKey();
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/35. Console - Numerical Data Types/NumericalDataTypesExample/bin/Debug/Numeric...
10
20
30
40
50
60
70
80
90.23
100.23489
110.882932
```

MinValue and MaxValue

NumericalDataType.MinValue

- This statement returns the minimum value of the specified numerical data type.
- Syntax: *datatype*.*MinValue*
- Ex: `int.MinValue`

NumericalDataType.MaxValue

- This statement returns the maximum value of the specified numerical data type.
- Syntax: *datatype*.*MaxValue*
- Ex: `int.MaxValue`

MinValue and MaxValue - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “`MinValue.MaxValueExample`”.
- Type the location as “`C:\CSharp`”.
- Type the solution name as “`MinValue.MaxValueExample`”.

- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //get the minimum value of "int" data type
        int min = int.MinValue;

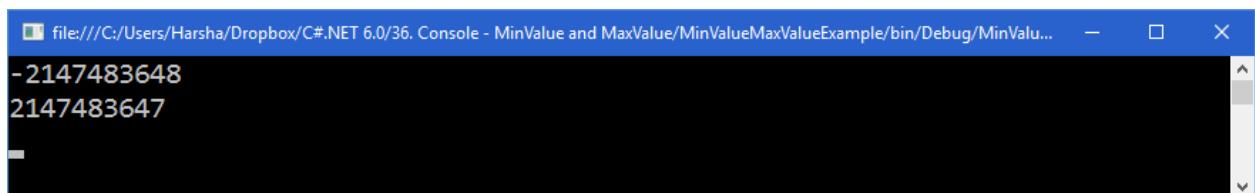
        //get the maximum value of "int" data type
        int max = int.MaxValue;

        //display the min and max values
        System.Console.WriteLine(min);
        System.Console.WriteLine(max);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/36. Console - MinValue and MaxValue/MinValueMaxValueExample/bin/Debug/MinValue...". The window contains the following text:
-2147483648
2147483647

char - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “charExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “charExample”.
- Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

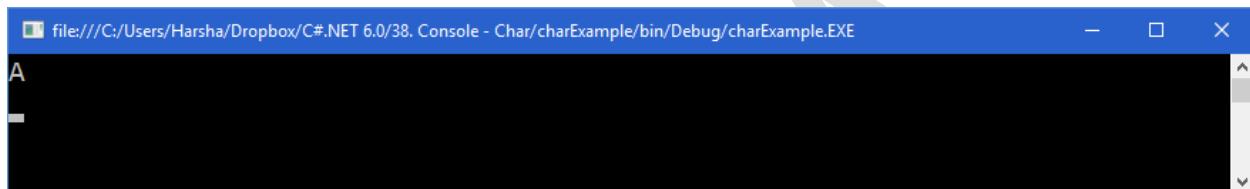
Program.cs

```
class Program
{
    static void Main()
    {
        //create variable of "char" data type
        char ch = 'A';
        //display the value of the variable
        System.Console.WriteLine(ch);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String

String

- String: A string is a group of characters.
- String literal should be written inside the double quotes.
- All the names are strings. Ex: person names, city names, country names etc.
- String may contain alphabets, numbers, spaces and also special symbols.
- Alpha-numerical values are also treated as strings. Ex: car number, phone number, bank ifsc codes etc.
- Syntax to create a string variable:
 - `string variablename = "value";`

String - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StringExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StringExample”. Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

Program.cs

```
class Program
{
    static void Main()
    {
        //create a variable
        string s = "Hello 123 $#&";

        //display the value of the variable
        System.Console.WriteLine(s);
        System.Console.WriteLine(s);
        System.Console.WriteLine(s);

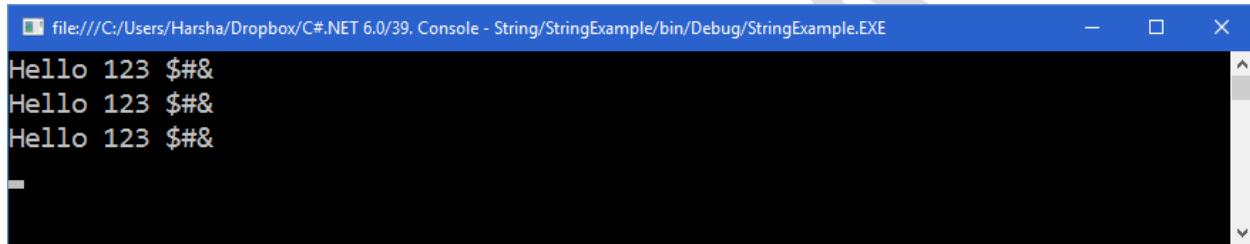
        System.Console.ReadKey();
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/39. Console - String/StringExample/bin/Debug/StringExample.EXE
Hello 123 $#&
Hello 123 $#&
Hello 123 $#&
```

Bool - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “boolExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “boolExample”.
- Click on OK.
- It shows “Program.cs” file automatically.
- Type the following code for “Program.cs” file:

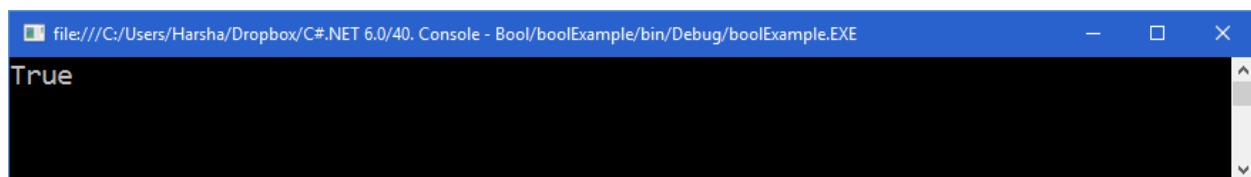
Program.cs

```
class Program
{
    static void Main()
    {
        //create variable of "bool" data type
        bool b = true;
        //display the value of the variable
        System.Console.WriteLine(b);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/40. Console - Bool/boolExample/bin/Debug/boolExample.EXE". The main area of the window displays the word "True".

Operators

What is Operator:

- Operator is a symbol to perform an operation.
- An operator receives one or two operands and perform some operation & returns the result.
- Types of operators:
 1. Arithmetical Operators
 2. Assignment Operators
 3. Increment and Decrement Operators
 4. Relational Operators
 5. Logical Operators
 6. Concatenation Operator
 7. Conditional Operator

Arithmetical Operators

Arithmetical Operators		
Sl. No	Operator	Description
1	+	Addition
2	-	Subtraction
3	*	Multiplication
4	/	Division
5	%	Remainder

Arithmetical Operators - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArithmeticalOperatorsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArithmeticalOperatorsExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create two variables of "double" data type
        double a = 10, b = 3;

        //addition
        double c = a + b;

        //subtraction
        double d = a - b;

        //multiplication
        double e = a * b;

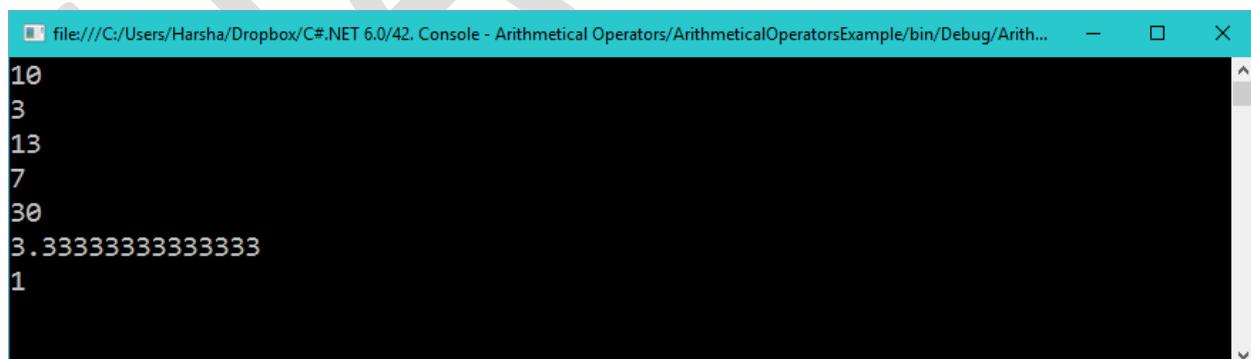
        //division
        double f = a / b;
    }
}
```

```
//remainder  
double g = a % b;  
  
//display all the values  
System.Console.WriteLine(a); //Output: 10  
System.Console.WriteLine(b); //Output: 3  
System.Console.WriteLine(c); //Output: 13  
System.Console.WriteLine(d); //Output: 7  
System.Console.WriteLine(e); //Output: 30  
System.Console.WriteLine(f); //Output: 3.33333333333  
System.Console.WriteLine(g); //Output: 1  
  
System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/42. Console - Arithmetical Operators/ArithmeticalOperatorsExample/bin/Debug/Arith...  
10  
3  
13  
7  
30  
3.33333333333  
1
```

Assignment Operators

Assignment Operators		
Sl. No	Operator	Description
1	=	Assigns to
2	+=	Add and assigns to
3	-=	Subtract and assigns to
4	*=	Multiply and assigns to
5	%=	Remainder and assigns to

Assignment Operators - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AssignmentOperatorsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AssignmentOperatorsExample”.
- Click on OK.

Program.cs

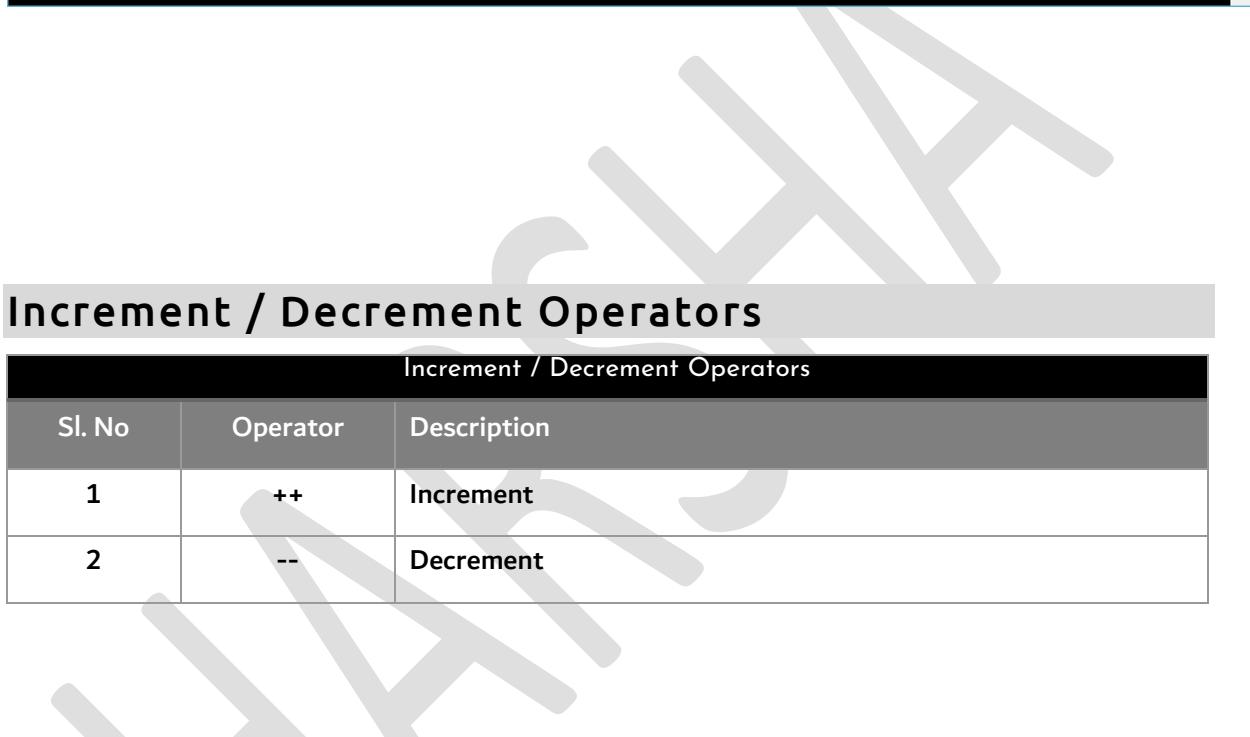
```
class Program
{
    static void Main()
```

```
{  
    //create two variables of "int" data type  
    int a = 100;  
    int b;  
    //get the value from "a" and set the same into "b"  
    b = a;  
    //display the values of "a" and "b"  
    System.Console.WriteLine(a); //Output: 100  
    System.Console.WriteLine(b); //Output: 100  
    //a = a + 10  
    a += 10;  
    System.Console.WriteLine(a); //Output: 110  
    //a = a - 10  
    a -= 10;  
    System.Console.WriteLine(a); //Output: 100  
    //a = a * 3  
    a *= 3;  
    System.Console.WriteLine(a); //Output: 300  
    //a = a / 3  
    a /= 3;  
    System.Console.WriteLine(a); //Output: 100  
    //a = a % 30  
    a %= 30;  
    System.Console.WriteLine(a); //Output: 10  
    System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/43. Console - Assignment Operators/AssignmentOperatorsExample/bin/Debug/Assign... - X
100
100
110
100
300
100
10
```

Increment / Decrement Operators

Increment / Decrement Operators		
Sl. No	Operator	Description
1	<code>++</code>	Increment
2	<code>--</code>	Decrement

Increment / Decrement Operators - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IncDecOperatorsExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “IncDecOperatorsExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create a variable of "int" data type
        int n = 10;
        //display the value of "n"
        System.Console.WriteLine(n); //Output: 10
        //n = n + 1
        n++;
        System.Console.WriteLine(n); //Output: 11

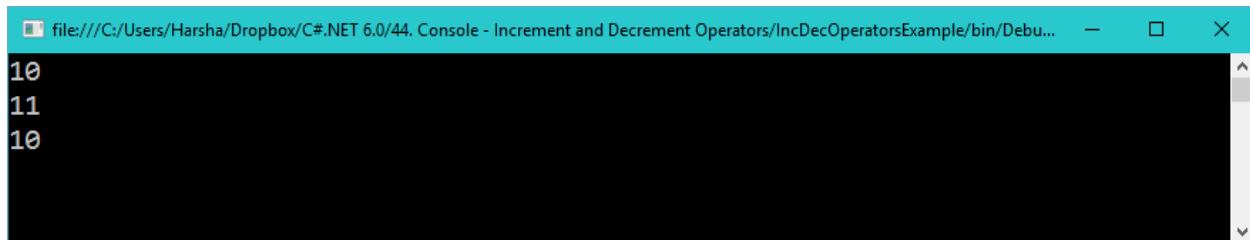
        //n = n - 1
        n--;
        System.Console.WriteLine(n); //Output: 10

        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/44. Console - Increment and Decrement Operators/IncDecOperatorsExample/bin/Debug/IncDecOperatorsExample.exe
10
11
10
```

Relational Operators

Relational Operators		
Sl. No	Operator	Description
1	==	Equal to
2	!=	Not equal to
3	<	Less than
4	>	Greater than
5	<=	Less than or equal to
6	>=	Greater than or equal to

Relational Operators - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “RelationalOperatorsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RelationalOperatorsExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create two variables of "long" data type
        long x = 1000, y = 2000;

        //check whether x and y are equal
        bool b1 = (x == y);

        //check whether x and y are not equal
        bool b2 = (x != y);

        //check whether x is less than y
        bool b3 = (x < y);

        //check whether x is greater than y
        bool b4 = (x > y);

        //check whether x is less than or equal to y
        bool b5 = (x <= y);

        //check whether x is greater than or equal to y
        bool b6 = (x >= y);

        System.Console.WriteLine(b1); //Output: false
        System.Console.WriteLine(b2); //Output: true
        System.Console.WriteLine(b3); //Output: true
```

```

System.Console.WriteLine(b4); //Output: false
System.Console.WriteLine(b5); //Output: true
System.Console.WriteLine(b6); //Output: false

System.Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window showing the output of the C# program. The window title is "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/45. Console - Relational Operators/RelationalOperatorsExample/bin/Debug/Relational...". The output text is:

```

False
True
True
False
True
False

```

Logical Operators

Logical Operators		
Sl. No	Operator	Description
1	&&	And (Both conditions must be true)
2		Or (Any one of the conditions must be true)
3	!	Not (Given condition will be reverse)

Logical Operators - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “LogicalOperatorsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LogicalOperatorsExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create 3 variables of "int" data type
        int a = 10, b = 20, c = 10;

        //and
        bool result1 = ((a == b) && (b > c));
        System.Console.WriteLine(result1); //Output: false

        //or
        bool result2 = ((a == b) || (b > c));
        System.Console.WriteLine(result2); //Output: true

        //not
    }
}
```

```

bool result3 = !(a == b);
System.Console.WriteLine(result3); //Output: true

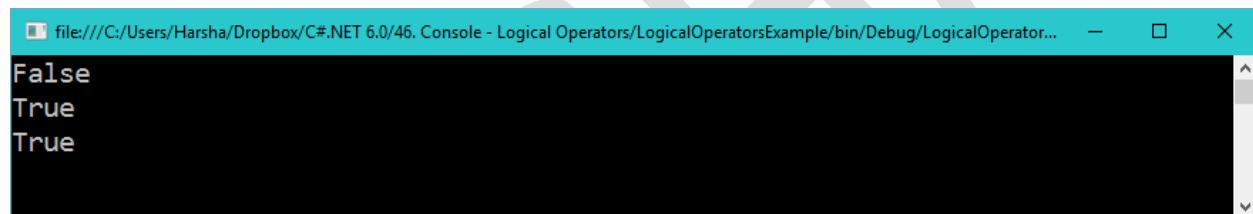
System.Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
False
True
True
```

Concatenation Operator

Concatenation Operator		
Sl. No	Operator	Description
1	+	<p>Concatenation. It attaches two values and returns as a string.</p> <p>The following cases, "+" operator is "concatenation operator".</p> <ul style="list-style-type: none"> • String + String • String + Number • Number + String

Concatenation Operator - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ConcatenationOperatorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ConcatenationOperatorExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create strings
        string s1 = "peers";
        string s2 = "tech";
        string s3;

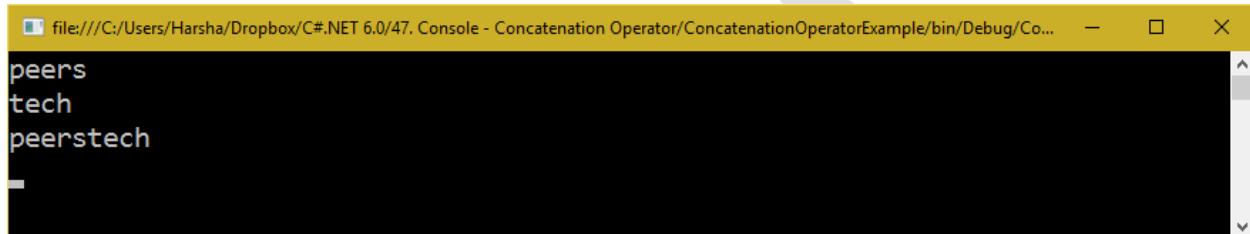
        //concatenate s1 and s2 and store the result in s3
        s3 = s1 + s2;

        System.Console.WriteLine(s1); //Output: peers
        System.Console.WriteLine(s2); //Output: tech
        System.Console.WriteLine(s3); //Output: peeerstech
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/47. Console - Concatenation Operator/ConcatenationOperatorExample/bin/Debug/Co...
peers
tech
peerstech
```

Conditional Operator

Conditional Operator		
Sl. No	Operator	Description
1	<code>(condition)? value1 : value2</code>	<p>Checks the given condition.</p> <p>Returns the value1, if given condition is TRUE</p> <p>Returns the value2, if the given condition is FALSE.</p>

Conditional Operator - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ConditionalOperatorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ConditionalOperatorExample”.
- Click on OK.

Program.cs

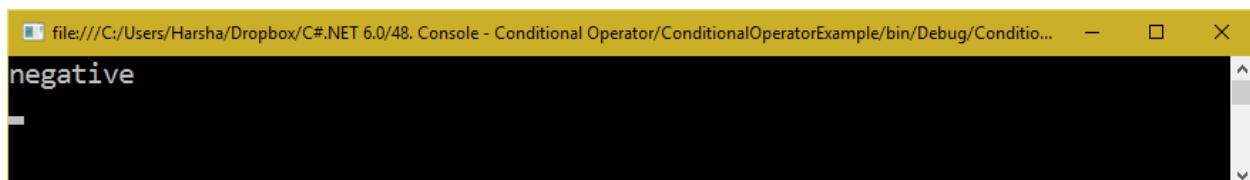
```
class Program
{
    static void Main()
    {
        //create a variable of "int" data type
        int n = -100;

        //check the condition; store "positive" if the condition is true; store
        "negative" if the condition is false
        string s = ( (n >= 0) ? "positive" : "negative");
        System.Console.WriteLine(s); //Output: negative
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar says "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/48. Console - Conditional Operator/ConditionalOperatorExample/bin/Debug/Conditi...". The main area of the window displays the word "negative" in white text on a black background.

Control Statements

What are control statements?

- Control statements are used to control the “program execution flow”.
- That means these are used to “go forward” or “go backward” within the program.

Types of control statements:

✧ Conditional Control Statements

- If
- Switch-case

✧ Looping Control Statements

- While
- Do-While
- For

✧ Jumping Control Statements

- Break
- Continue
- Goto

If

If			
Sl. No	Control Statement	Syntax	Description
1	If	<pre>if (condition) { Your code here }</pre>	<p>It executes the “if block” only if the given condition is TRUE.</p> <p>It doesn’t execute anything if the given condition is FALSE.</p>
2	If-Else	<pre>if (condition) { Your code here } else { Your code here }</pre>	<p>It executes the “if block” if the given condition is TRUE.</p> <p>It executes the “else block” if the given condition is FALSE.</p>
3	Else-If	<pre>if (condition) { Your code here } else if (condition) { Your code here } else if (condition) { Your code here } else { Your code here }</pre>	<p>It executes the “if block”, if the “first condition” is TRUE.</p> <p>It executes the “first else if block”, if the second condition is TRUE.</p> <p>It executes the “second else if block”, if the third condition is TRUE.</p> <p>It executes the “else block” if all the conditions are FALSE.</p>
4	Nested If	<pre>if (condition) {</pre>	If inside another if.

```
if (condition)
{
    Your code here
}
else
{
    Your code here
}
else
{
    if (condition)
    {
        Your code here
    }
    else
    {
        Your code here
    }
}
```

If - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IfExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IfExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create variables
        int n = 100;

        //check whether n is equal to 100
        if (n == 100)
        {
            System.Console.WriteLine("n is equal to 100");
        }
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
n is equal to 100
```

If – Else - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IfElseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IfElseExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create variables
```

```
int n = 150;

//check whether n is equal to 100
if (n == 100)
{
    System.Console.WriteLine("n is equal to 100");
}
else
{
    System.Console.WriteLine("n is not equal to 100");
}
System.Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
n is not equal to 100
```

Else – If - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ElseIfExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ElseIfExample”.
- Click on OK.

Program.cs

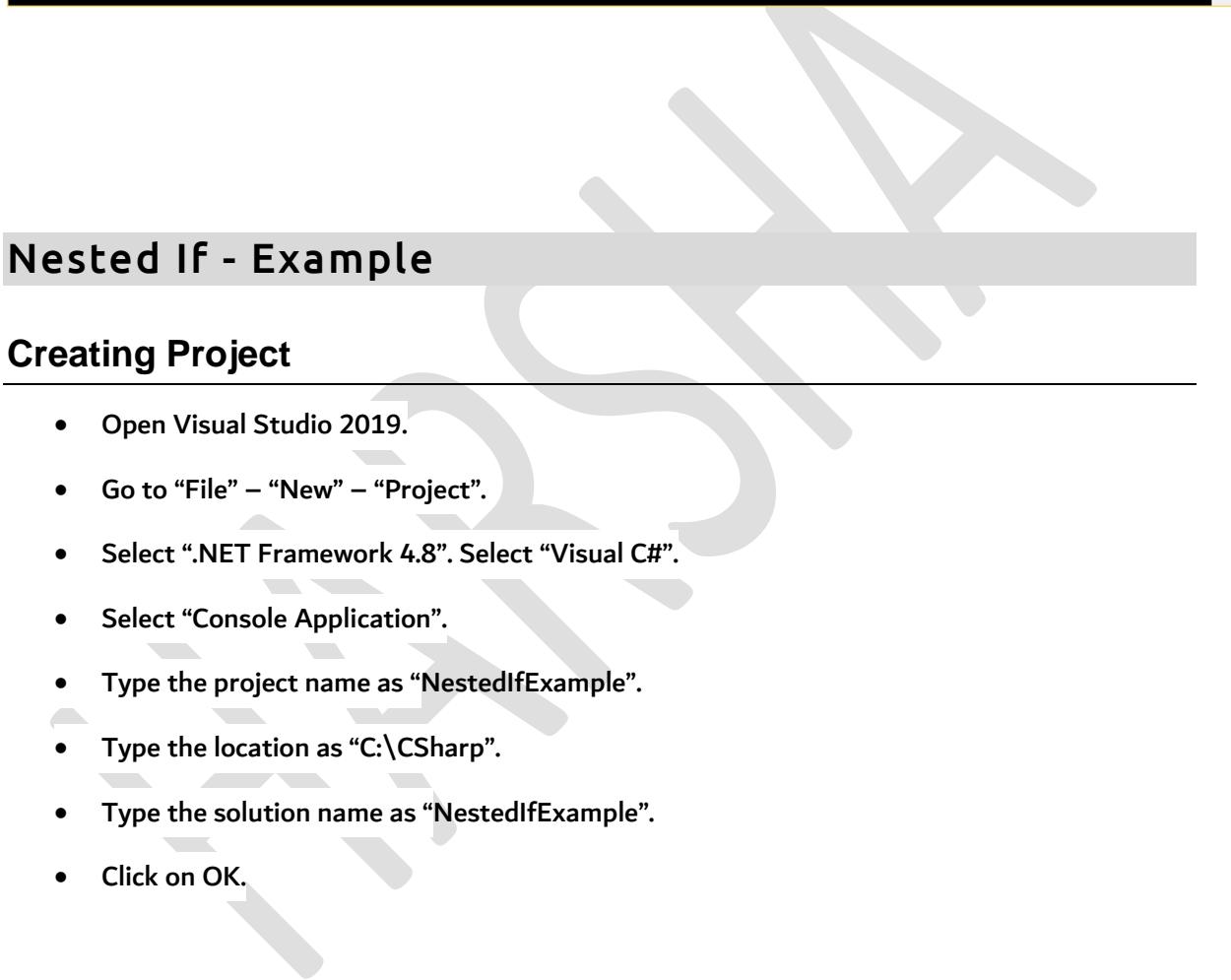
```
class Program
{
    static void Main()
    {
        //create variables
        int a = 10, b = 20;

        //check which is the big number
        if (a == b)
            System.Console.WriteLine("a and b are equal");
        else if (a > b)
            System.Console.WriteLine("a is bigger than b");
        else
            System.Console.WriteLine("b is bigger than a");
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/52. Console - Else - If/ElseIfExample/bin/Debug/ElseIfExample.EXE
b is bigger than a
```

Nested If - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “NestedIfExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NestedIfExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create variables
        int a = 150, b = 100;
```

```

string msg;

//outer if
if (a >= b)
{
    //inner if
    if (a > b)
    {
        msg = "a is greater than b";
    }
    //"else" for "inner if"
    else
    {
        msg = "a is equal to b";
    }
}
//"else" for "outer if"
else
{
    msg = "a is less than b";
}
System.Console.WriteLine(msg); //Output: a is greater than b
System.Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/53. Console - Nested If/NestedIfExample/bin/Debug/NestedIfExample.EXE
a is greater than b

```

Switch – Case

Switch-case			
Sl. No	Control Statement	Syntax	Description
1	Switch-case	<pre>switch (variable) { case value1: your code here; break; case value2: your code here; break; ... default: your code here; break; }</pre>	<p>It checks the variable's value whether it matches with which case & executes the corresponding code.</p> <p>It executes the default code if all cases are not matched.</p>

Switch – Case - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SwitchCaseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SwitchCaseExample”.
- Click on OK.

Program.cs

```
class Program
{
```

```
static void Main()
{
    //create variables
    int monthnumber = 7;
    string monthname;

    //check the value of monthnumber whether it matches with any one of
    //the following cases
    switch (monthnumber)
    {
        case 1: monthname = "Jan"; break;
        case 2: monthname = "Feb"; break;
        case 3: monthname = "Mar"; break;
        case 4: monthname = "Apr"; break;
        case 5: monthname = "May"; break;
        case 6: monthname = "Jun"; break;
        case 7: monthname = "Jul"; break;
        case 8: monthname = "Aug"; break;
        case 9: monthname = "Sep"; break;
        case 10: monthname = "Oct"; break;
        case 11: monthname = "Nov"; break;
        case 12: monthname = "Dec"; break;
        default: monthname = "unknown"; break;
    }
    System.Console.WriteLine(monthname); //Output: Jul
    System.Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/54. Console - Switch-case/SwitchCaseExample/bin/Debug/SwitchCaseExample.EXE
Jul
```

While

while			
Sl. No	Control Statement	Syntax	Description
1	While	while (<i>condition</i>) { <i>your code here</i> }	<p>It executes the code repeatedly, while the condition is TRUE.</p> <p>If exits from the loop automatically when the condition is FALSE.</p>

While - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “WhileExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WhileExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //initialization
        int i = 1;
        //condition
        while (i <= 10)
        {
            System.Console.WriteLine(i);
            i++; //incrementation
        }
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1
2
3
4
5
6
7
8
9
10
```

Do – While

do-while			
Sl. No	Control Statement	Syntax	Description
1	Do-While	<pre>do { <i>Your code here</i> } while(condition);</pre>	<p>It repeats the code execution while the condition is TRUE.</p> <p>It executes the code at least once. That means it will not check the condition for the first iteration.</p> <p>If exit from the loop automatically when the condition is FALSE.</p>

Do – While - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DoWhileExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DoWhileExample”.
- Click on OK.

Program.cs

```
class Program
{
```

```
static void Main()
{
    //initialization
    int i = 1;
    do
    {
        System.Console.WriteLine(i);
        i++; //incrementation
    } while (i <= 10); //condition
    System.Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1
2
3
4
5
6
7
8
9
10
```

for

for			
Sl. No	Control Statement	Syntax	Description
1	for	<pre><code>for (initialization; condition; iteration) { Your code here }</code></pre>	Initialization, condition, iteration (increment or decrement) are written in a single line; so that it is easy to understand.

for - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ForExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ForExample”.
- Click on OK.

Program.cs

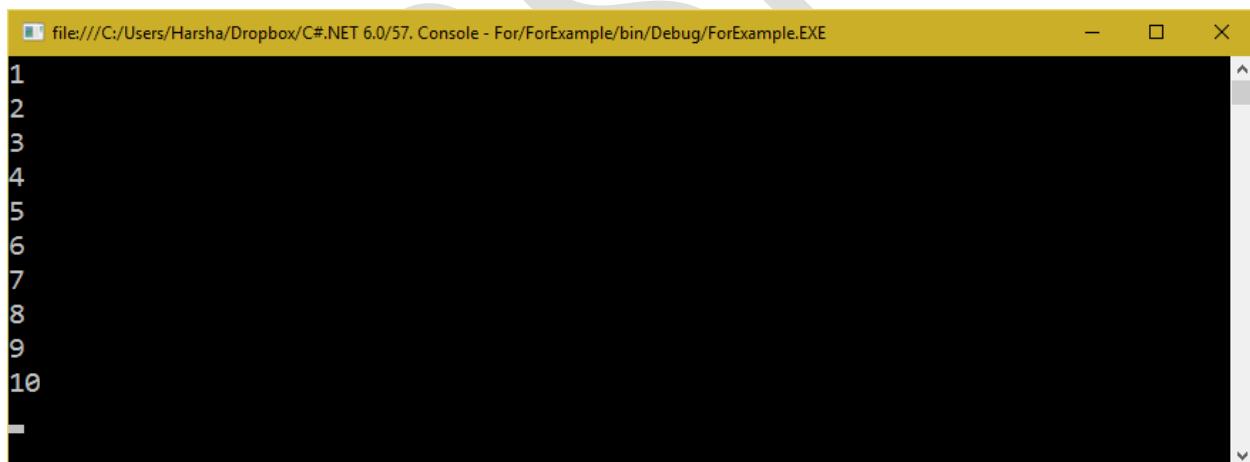
```
class Program
{
```

```
static void Main()
{
    //initialization; condition; incrementation
    for (int i = 1; i <= 10; i++)
    {
        System.Console.WriteLine(i);
    }
    System.Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1
2
3
4
5
6
7
8
9
10
```

Break

Sl. No	Control Statement	Syntax	Description
1	Break	<code>break;</code>	It terminates (stops) the loop.

Break - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “BreakExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “BreakExample”. Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //initialization; condition; incrementation
        for (int i = 1; i <= 10; i++)
        {
            System.Console.WriteLine(i);
            if (i == 6)
                break; //stop the loop when "i" value is reached to "6".
        }
    }
}
```

```
        }
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1
2
3
4
5
6
```

Continue

continue			
Sl. No	Control Statement	Syntax	Description
1	Continue	continue;	It skips the current iteration and jumps to the next iteration.

Continue - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ContinueExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ContinueExample”. Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //initialization; condition; incrementation
        for (int i = 1; i <= 10; i++)
        {
            if (i == 6)
                continue; //skip "6" and go to "7"

            System.Console.WriteLine(i);
        }
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
1
2
3
4
5
7
8
9
10
```

Goto

goto			
Sl. No	Control Statement	Syntax	Description
1	Goto	<code>goto LabelName;</code>	It jumps to the specified label, within the same method.

Goto - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “GotoExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “GotoExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        System.Console.WriteLine("one");
        System.Console.WriteLine("two");

        //jump to mylabel
        goto mylabel;

        System.Console.WriteLine("three");
        System.Console.WriteLine("four");
        System.Console.WriteLine("five");

        //mylabel starts here
        mylabel:
        System.Console.WriteLine("six");
        System.Console.WriteLine("seven");

        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/60. Console - Goto/GotoExample/bin/Debug/GotoExample.EXE
one
two
six
seven
```

Nested For Loops

Syntax of Nested For Loop	Description
<pre>for (initialization; condition; iteration) { for (initialization; condition; iteration) { Your code here } }</pre>	The “inner for loop” executes ‘n’ no. of times, while the condition outer for loop’s condition is true.

Nested For Loops - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “NestedForLoopsExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “NestedForLoopsExample”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //outer loop (5 times)
        for (int i = 1; i <= 5; i++)
        {
            //inner loop (10 times)
            for (int j = 1; j <= 10; j++)
            {
                System.Console.Write(j);
                System.Console.Write(", ");
            }
            System.Console.WriteLine();
        }
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/61. Console - Nested For Loops/NestedForLoopsExample/bin/Debug/NestedForLoopsE... — X
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

C#.NET – Object Oriented Programming (OOP)

Introduction to Object Oriented Programming

Types of Programming Languages

1. Imperative Programming Languages:

- The programs are written as a series of statements that execute in top-to-bottom approach.
- Ex: BASIC, Assembly languages

2. Structured Programming Languages:

- These are also imperative programming languages with control structures like if, switch-case, while, do-while, for, break, continue, return etc.
- Ex: Cobol, Pascal

3. Procedural Programming Languages:

- These are also structured programming languages with procedures (functions) concept.
- A procedure or function is a collection statements present independently in the program and can be called many times during the normal flow of execution.
- Ex: C

4. Object Oriented Programming Languages:

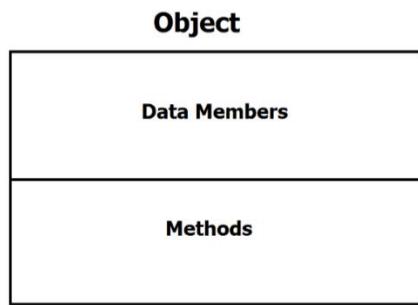
- These are also procedural programming languages with objects and classes.
- Object is a physical item; Class is a model of object.
- Ex: C++, Java, C#.NET

Introduction of Object Oriented Programming (OOP)

- “Object oriented programming” is a programming style / programming paradigm, which provides necessary guidelines for the programmers to write the programs in a well-organized, understandable, secured, re-usable, efficient, structured, clean, maintainable manner.
- OOP is followed in almost-all modern programming languages such as C++, VC++, Java, C#.NET, VB.NET etc.
- .NET developers must know OOP.
- Advantages of OOP:
 1. Modularity: Dividing large programs into multiple parts called classes.
 2. Re-usability: The class created once can be called many times.
 3. Security: The members of a class can be private or public. The private members can't be accessible outside the class.

Object

- An object represents a real world item. For example, you are an object, your laptop is an object, and your city is an object.
- An object consists of "details" and "functionality".
- Details are called as "fields".
- Functionality is called as "functions" or "methods" or "operations".
- All the objects will be stored in an area called "heap" in RAM.



Fields

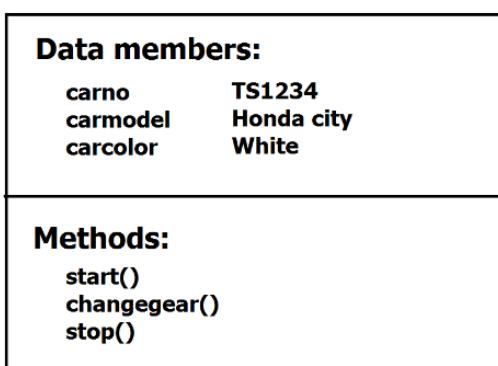
- Fields also called as "details".
- The variables inside object are called as "Fields".
- Fields are used to store data.

Methods

- Methods also called as "functionality" or "functions" or "operations".
- A method is a "set of statements" to do a particular task in the program.
- Methods are used to manipulate Fields.

Example:

Object "mycar"

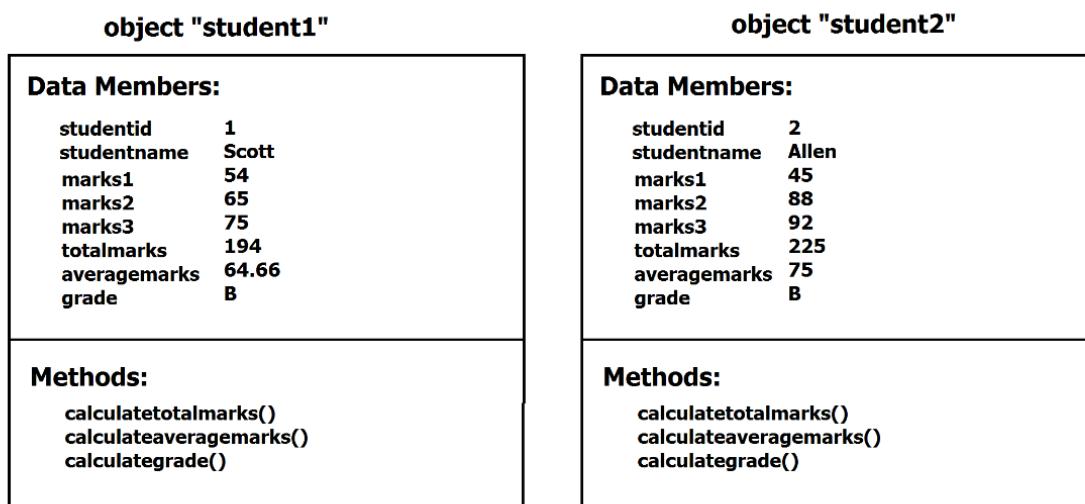
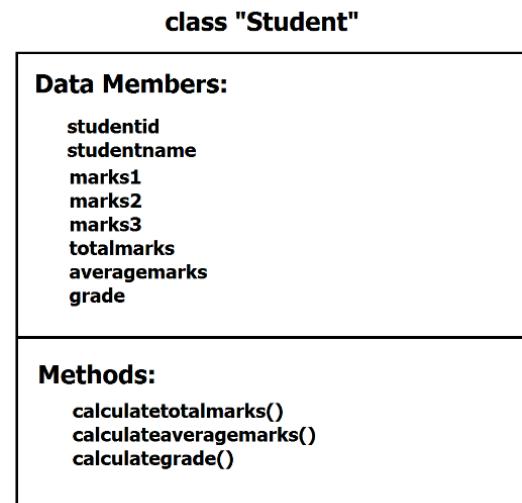


Class

- A class is a "model" or "type" or "blue-print" of object.
- Class is used to define the common members of similar objects.
 - ✧ For example, there are 10 students. Each student is an object. So 10 students are 10 objects. In each student, we want to store "studentid", "studentname", "marks". So first we have to create a class called "Student" and we have to create "studentid", "studentname", "marks" as members of the "Student" class. Then we have to create objects based on the class.
- A class specifies the list of Fields and methods that you want to store in every object.
- For a class, memory will not be allocated.
- So you must create an object for the class, in order to access its members.
- Class is a just "model" only; no actual data will be stored in class. Object is the "real instance"; actual data will be stored in object only.
- Based on a class, any no. of objects can be created.
- When an object is created, the following process happens.
 1. Memory will be allocated in RAM (Random Access Memory) for the object.
 2. All the Fields of the class will be stored in the object.
 3. Fields of the object will be initialized with default values.
 - Numerical Fields: 0
 - Char / string: null
 - Bool: false
 - DateTime: 1/1/0001 12:00:00 AM
- All the objects are stored in RAM (temporarily). Objects are created when the program execution starts. Objects are automatically deleted (erased from memory) when the program execution ends.

- Each object's memory will be allocated separately in RAM.

Example:



Reference variables

- A “reference variable” stores address of an object of same class.
- You should store the “address of an object” in the reference variable; through the reference variable only, we can access the object, in further statements.
- All reference variables will be stored in an area called “stack” in RAM.

Syntax of creating class

```
class Classname  
{  
    Class members here  
}
```

Syntax of creating a Field in a class:

```
accessmodifier qualifier datatype Fieldname = value;
```

Syntax of creating a method in a class:

```
accessmodifier qualifier returntype methodname(arguments)  
{  
    Code here  
}
```

Syntax of creating reference variable

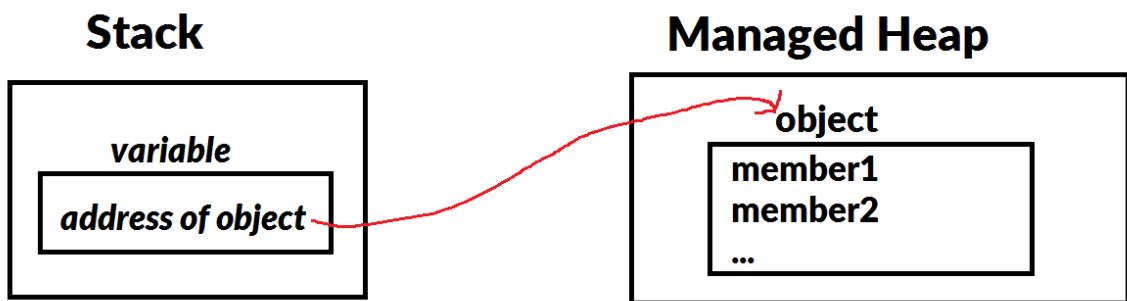
Classname Referencevariablename;

Syntax of creating object (in heap):

new *Classname ()*;

Memory allocation of Objects

- All the reference variables are stored in an area called "stack" in RAM. For every method call, a "stack" will be created automatically.
- All the objects are stored in an area called "heap" in RAM. For entire application, a "heap" will be created.
- You must store the address of an object into the reference variable; otherwise you can't access the object.



Principles of Object Oriented Programming

Principles of OOP

- OOP following the following principles.

1) Encapsulation

2) Abstraction

3) Inheritance

4) Polymorphism

Encapsulation:

- Encapsulation is a concept of combining Fields and methods as a single unit called “object”.
- This is implemented by creating classes with Fields and methods.
- Fields are variables that stores the data; Methods are functions that manipulates the data.

Abstraction:

- Abstraction is a concept of hiding implementation details (internal logic) and providing only necessary details to other classes.
- This is implemented by creating private Fields and public methods.

Inheritance:

- Inheritance is a concept of creating “parent-child” relationship among two or more classes.

- As a result of inheritance, all the members of parent class are accessible in child class.

Polymorphism:

- Polymorphism means "decision making".
- Polymorphism means "the ability of a statement that calls different methods at different situations".

Types of polymorphism:

A. Static polymorphism (or) Compile-time polymorphism (or) Early binding:

- Static polymorphism means method overloading.
- Method overloading is a concept of creating multiple methods with same name in the same class with different types of arguments. While calling the method, the matching method will be executed, depending on the arguments that are passed.
- The decision of which method is to be executed will be taken at compilation time (by c#.net compiler). That's why it is called "Compile-time polymorphism".
- Example of static polymorphism: Method overloading

B. Dynamic polymorphism (or) Run-time polymorphism (or) Late binding:

- Dynamic polymorphism means method overriding.
- Method overriding is a concept of creating two methods with same name and same signature; one is in parent class and other one is in child class; both methods have different code.

- You will create a reference variable for the parent class (or) interface; assign the reference of child class's object; and call the method. Then the current child class's method will be called.
- The decision of which class's method is to be executed will be taken at run time (by .net CLR). That's why it is called "Run-time polymorphism".
- Example of dynamic polymorphism: Reference variable of interface type.

Access Modifiers

Access Modifiers

- Access Modifiers are also called as "Access Specifiers", which are used to specify the access privileges of a member of a class.
- Access Modifiers can be applicable to all types of members (such as Field, method, constructor, property etc.) that tell which classes can access the member and which can't.
- Access Modifiers are used to create security for the member of a class.
- List of access modifiers in c#.net:
 1. private (default)
 2. protected
 3. internal
 4. protected internal
 5. public

1. **private:** The private members are accessible “only within the same class”. These are not accessible in any other classes. “Private” is the default access modifier in c#.net. That means, if you don’t specify any access modifier, by default, “private” will be applied.

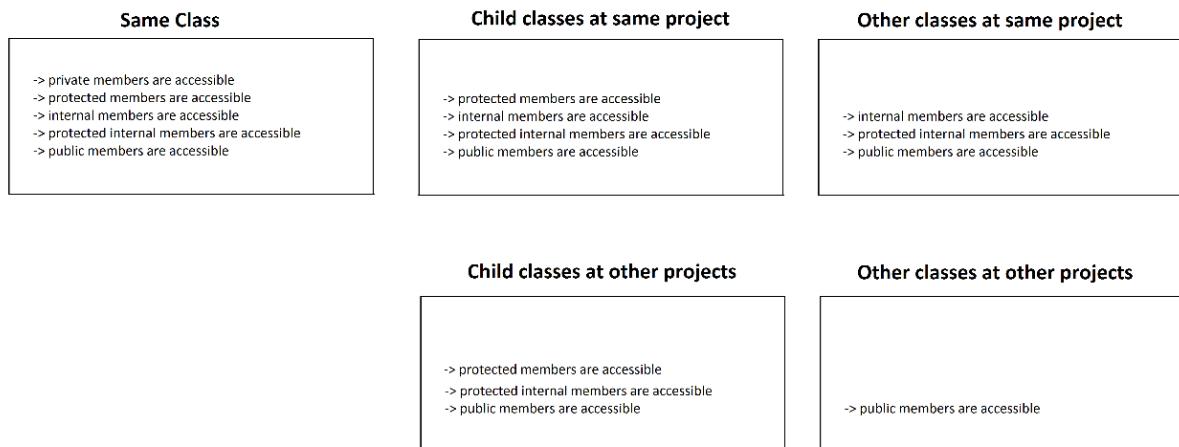
2. **protected:** The protected members are accessible “within the same class” and also within the “child classes at same project” and “child classes at other projects”. These are not accessible in any other classes. Note: The other projects must add the reference of current project.

3. **internal:** The internal members are accessible “anywhere within the same project”. These are not accessible in any classes at other projects.

4. **protected internal:** “Protected internal” is a combination of “protected” and “internal”. The protected internal members are accessible “anywhere within the project” and also accessible “within the child classes at other projects”. These are not accessible in other classes at other projects. Note: The other projects must add the reference of current project.

5. **public:** The public members are accessible “everywhere”.

Access Modifiers



Access Modifiers

Access Modifier	In the same class	In the child classes at same project	In the other classes at the same project	Child classes at other projects	Other classes at other projects
private	Yes	No	No	No	No
protected	Yes	Yes	No	Yes	No
internal	Yes	Yes	Yes	No	No
protected internal	Yes	Yes	Yes	Yes	No
public	Yes	Yes	Yes	Yes	Yes

Access Modifiers - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AccessModifiersExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AccessModifiersExample”. Click on OK.

Program.cs

```
namespace AccessModifiersExample
{
    //same class
    class Class1
    {
        private int a; //private member
        protected int b; //protected member
        internal int c; //internal member
        protected internal int d; //protected internal member
        public int e; //public member

        public void Method1()
        {
            a = 10; //private member is accessible in the same class
            b = 20; //protected member is accessible in the same class
            c = 30; //internal member is accessible in the same class
            d = 40; //protected internal member is accessible in the same class
            e = 50; //public member is accessible in the same class
        }
}
```

```
}
```

//child class in the same project

```
class Class2: Class1
{
    public void Method2()
    {
        b = 20; //protected member is accessible in the child class at same
project
        c = 30; //internal member is accessible in the child class at same
project
        d = 40; //protected internal member is accessible in the child class at
same project
        e = 50; //public member is accessible in the child class at same project
    }
}
```

//other class in the same project

```
class Class3
{
    public void Method3()
    {
        Class1 c1 = new Class1();
        c1.c = 30; //internal member is accessible in the other class at same
project
        c1.d = 40; //protected internal member is accessible in the other class
at same project
        c1.e = 50; //public member is accessible in the other class at same
project
    }
}
```

//other class in the same project

```
class Program
{
    static void Main()
    {
```

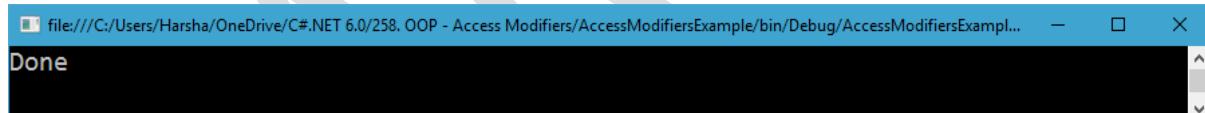
```
Class1 c1 = new Class1();
c1.c = 30; //internal member is accessible in the other class at same
project
c1.d = 40; //protected internal member is accessible in the other class
at same project
c1.e = 50; //public member is accessible in the other class at same
project

System.Console.WriteLine("Done");
System.Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Access Modifiers for classes

Access Modifiers for Classes

- Access Modifiers can be used for classes also, to specify where the class is accessible.

- The following access modifiers can be used for classes:

- List of access modifiers for classes in c#.net:

1. internal (default)

2. public

3. private

1. internal: The internal classes can be accessible within the same project only. They are not accessible in other projects. “Internal” is the default access modifier for the classes.

2. public: The public classes can be accessible anywhere (within the same project and also in other projects too).

3. private: Only inner classes can be “private classes”. The “private inner classes” can be accessible within the same outer classes only. They are not accessible in other classes.

Object Oriented Programming – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “OOPExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OOPExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public int a, b;
}

class Program
{
    static void Main()
    {
        Sample s1; //create reference variable
        s1 = new Sample(); //create object
        s1.a = 10;
        s1.b = 20;

        Sample s2; //create reference variable
        s2 = new Sample(); //create object
        s2.a = 30;
        s2.b = 40;

        //display values
        System.Console.WriteLine(s1.a); //Output: 10
        System.Console.WriteLine(s1.b); //Output: 20
        System.Console.WriteLine(s2.a); //Output: 30
        System.Console.WriteLine(s2.b); //Output: 40

        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
10
20
30
30
```

Object Oriented Programming – Student Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “OOPStudentExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OOPStudentExample”.
- Click on OK.

Program.cs

```
class Student
{
    public int StudentId;
    public string StudentName;
    public int Marks;
```

```
}

class Program
{
    static void Main()
    {
        //create reference variables
        Student s1, s2;

        //create objects
        s1 = new Student();
        s2 = new Student();

        //set data into first object
        System.Console.Write("Enter first student id: ");
        s1.StudentId = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter first student name: ");
        s1.StudentName = System.Console.ReadLine();
        System.Console.Write("Enter first student marks: ");
        s1.Marks = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.WriteLine();

        //set data into second object
        System.Console.Write("Enter second student id: ");
        s2.StudentId = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter second student name: ");
        s2.StudentName = System.Console.ReadLine();
        System.Console.Write("Enter second student marks: ");
        s2.Marks = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.WriteLine();

        //get data from first object
        System.Console.WriteLine("Student ID: " + s1.StudentId);
        System.Console.WriteLine("Student Name: " + s1.StudentName);
        System.Console.WriteLine("Marks: " + s1.Marks);
        System.Console.WriteLine("-----");

        //get data from second object
        System.Console.WriteLine("Student ID: " + s2.StudentId);
        System.Console.WriteLine("Student Name: " + s2.StudentName);
        System.Console.WriteLine("Marks: " + s2.Marks);
        System.Console.WriteLine("-----");

        System.Console.ReadKey();
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
C:\Users\Harsha\Google Drive\C#.NET 7.0\045. OOP - Student Example\OOPStudentExample\OOPStudentExample\bin\Debug\OOPStudentExample.exe
Enter first student id: 1
Enter first student name: Scott
Enter first student marks: 75

Enter second student id: 2
Enter second student name: Smith
Enter second student marks: 45

Student ID: 1
Student Name: Scott
Marks: 75
-----
Student ID: 2
Student Name: Smith
Marks: 45
```

Object Oriented Programming – Employee Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “OPEEmployeeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OPEEmployeeExample”.
- Click on OK.

Program.cs

```
class Employee
{
    public int EmployeeId;
    public string EmployeeName;
    public int Salary;
}

class Program
{
    static void Main()
    {
        //create reference variables
        Employee s1, s2;

        //create objects
        s1 = new Employee();
        s2 = new Employee();

        //set data into first object
        System.Console.Write("Enter first Employee id: ");
        s1.EmployeeId = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter first Employee name: ");
        s1.EmployeeName = System.Console.ReadLine();
        System.Console.Write("Enter first Employee Salary: ");
        s1.Salary = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.WriteLine();

        //set data into second object
        System.Console.Write("Enter second Employee id: ");
        s2.EmployeeId = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter second Employee name: ");
        s2.EmployeeName = System.Console.ReadLine();
        System.Console.Write("Enter second Employee Salary: ");
        s2.Salary = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.WriteLine();

        //get data from first object
        System.Console.WriteLine("Employee ID: " + s1.EmployeeId);
        System.Console.WriteLine("Employee Name: " + s1.EmployeeName);
```

```
System.Console.WriteLine("Salary: " + s1.Salary);
System.Console.WriteLine("-----");
//get data from second object
System.Console.WriteLine("Employee ID: " + s2.EmployeeId);
System.Console.WriteLine("Employee Name: " + s2.EmployeeName);
System.Console.WriteLine("Salary: " + s2.Salary);
System.Console.WriteLine("-----");

System.Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
C:\Users\Harsha\Google Drive\C#.NET 7.0\045.1. OOP - Employee Example\OOPEmployeeExample\bin\Debug\OOPEmployeeExample.exe
Enter first Employee id: 101
Enter first Employee name: John
Enter first Employee Salary: 9500

Enter second Employee id: 102
Enter second Employee name: Jones
Enter second Employee Salary: 4570

Employee ID: 101
Employee Name: John
Salary: 9500
-----
Employee ID: 102
Employee Name: Jones
Salary: 4570
-----
```

Static Fields

- Static Fields are used to common data that belongs to all the objects. Non-Static Fields are stored in the objects; Static Fields are stored outside the objects. Static members are not accessible with reference variable, but accessible with “class name”.
- Fields are two types:
 1. Non-static Fields (or) Instance Fields
 2. Static Fields (or) Shared Fields

Sl. No	Non-Static Fields	Static Fields
1	By default, all the Fields are non-static Fields. These are not created with a keyword called “static”.	Static Fields are created using “static” keyword.
2	Non static Fields are stored in the object. That means when you create an object for the class, the memory will be allocated for all the non-static Fields in the object.	For static Fields, memory will not be allocated in the object. When you access the class name for the first time in the main method, then memory will be allocated for the static Fields.
3	Non-static Field's memory will be allocated separately for every object of the class.	Static Field's memory will be allocated only once for entire project.
4	Non-static Fields accessible with reference variable.	Static Fields are accessible with class.
5	If you don't create an object for the class, no memory will be allocated for non-static Field.	If you don't access the class in the entire program, no memory will be allocated for static Fields.
6	Use non-static Fields if you want to store object-specific data (which is different for each object).	Use static Fields if you want to store common data that belongs to all the objects.

Static Fields - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StaticFieldsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StaticFieldsExample”.
- Click on OK.

Program.cs

```
using System;

namespace StaticFieldsExample
{
    //creating a class called "Student".
    class Student
    {
        //instance Fields (or) non-static Fields
        public int studentid;
        public string studentname;
        public int marks;

        //static Field (or) shared Field
        public static string collegename;
    }

    class Program
    {
```

```

static void Main()
{
    //create two reference variables
    Student s1;
    Student s2;

    //create two objects
    s1 = new Student();
    s2 = new Student();

    //set data into members of s1
    s1.studentid = 1;
    s1.studentname = "scott";
    s1.marks = 70;

    //set data into members of s2
    s2.studentid = 2;
    s2.studentname = "allen";
    s2.marks = 80;

    //get data from members of s1
    Console.WriteLine("Student ID: " + s1.studentid);
    Console.WriteLine("Student Name: " + s1.studentname);
    Console.WriteLine("Marks: " + s1.marks);
    Console.WriteLine();

    //get data from members of s2
    Console.WriteLine("Student ID: " + s2.studentid);
    Console.WriteLine("Student Name: " + s2.studentname);
    Console.WriteLine("Marks: " + s2.marks);
    Console.WriteLine();

    //set data into static Field
    Student.collegename = "ABC college of technology";

    //get data from static Field
    Console.WriteLine("College Name: " + Student.collegename);

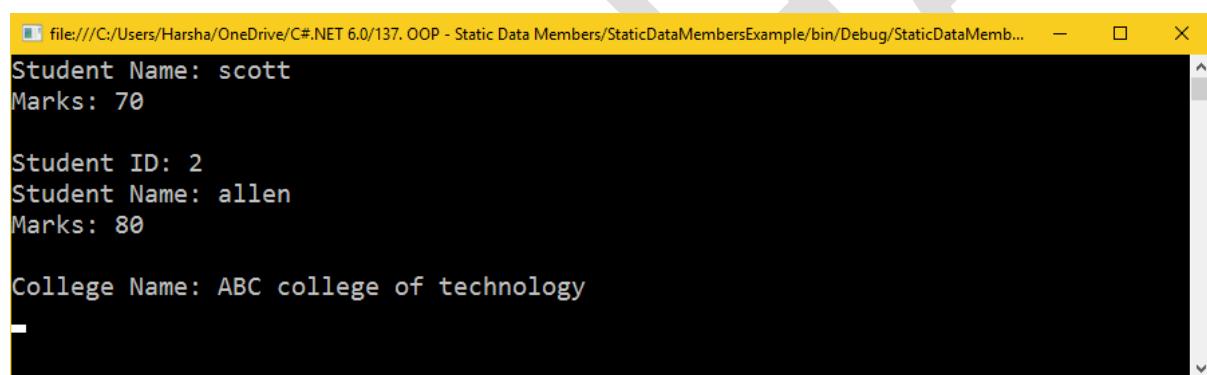
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/137. OOP - Static Data Members/StaticDataMembersExample/bin/Debug/StaticDataMemb...
Student Name: scott
Marks: 70

Student ID: 2
Student Name: allen
Marks: 80

College Name: ABC college of technology
```

Constant Fields

Constant Fields

- Constant Field's value can't be changed in the rest life of the object.
- Constant Fields must be initialized along with the declaration. Those can't be initialized in the constructor.
- Constant Fields are created using "const" keyword.
- Constant Fields are by default "static"; so they are accessible without creating an object for the class, using class name.
- Syntax:

Accessmodifier const Datatype Constantname = value;

Constant Fields - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Console Application".
- Type the project name as "ConstantsExample".
- Type the location as "C:\CSharp".
- Type the solution name as "ConstantsExample". Click on OK.

Program.cs

```
class Sample
```

```
{  
    public const int NoOfMonthsInYear = 12;  
}  
  
class Program  
{  
    static void Main()  
    {  
        System.Console.WriteLine(Sample.NoOfMonthsInYear);  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



ReadOnly Fields

Read-only Fields

- Read-only Fields must be initialized along with the declaration or in the constructor. It must be initialized before constructor ends.
- Read-only Field's value can't be changed in the rest life of the object.
- Read-only Fields are created using "readonly" keyword.
- Syntax:

Accessmodifier readonly Datatype Fieldname = Value;

Constant Fields (vs) Readonly Fields

Sl. No	Constant Field	Readonly Field
1	Declared with "const" keyword.	Declared with "readonly" keyword.
2	Can be created in a method or as a Field.	Can be created as a Field only.
3	Should be initialized along with its declaration.	Should be initialized either along with its declaration or in the constructor.
4	Useful to initialize a fixed value only.	Useful to initialize fixed value or result of a calculation.

ReadOnly Fields - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReadOnlyExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “ReadOnlyExample”. Click on OK.

Program.cs

```
class Sample
{
    public readonly int NoOfMonthsInYear = 12;
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        System.Console.WriteLine(s.NoOfMonthsInYear);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Methods

Methods

- The method is a “function” inside the class.
- The method is a set of instructions (statements) to perform a particular task.
- The method is a “re-usable code block”.
- Methods are used to divide the large code as small units.
- Methods improve the understandability of the program.
- Every method should have a name.
- Methods can be called any no. of times.
- Methods can be created and called in any order.
- Arguments: The value that are passed from calling portion to the method definition are called arguments or parameters. The data type of argument is called “argument type”. Every argument can be different type.
- Return Value: The value that is passed from method definition to the calling portion is called “return value”. A method can return only one value maximum. The data type of return value is called “return type”. If a method doesn’t return any value, its return type should be said as “void”.

Syntax of creating a method

```
AccessModifier Qualifier ReturnType Methodname(DataType1  
ArgumentVariable1, DataType2 ArgumentVariable2, ...)  
{  
    Your code here  
    return (value);  
}
```

Methods – Simple Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodsSimpleExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodsSimpleExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public void Address()
    {
        System.Console.WriteLine("Ameerpet");
        System.Console.WriteLine("Hyderabad");
        System.Console.WriteLine("India");
    }
}

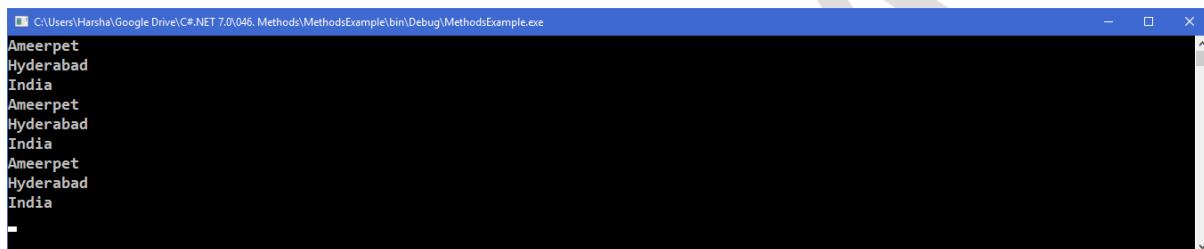
class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        s.Address();
        s.Address();
        s.Address();
        System.Console.ReadKey();
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
C:\Users\Harsha\Google Drive\C#.NET 7.0\046.Methods\MethodsExample\bin\Debug\MethodsExample.exe
Amerpet
Hyderabad
India
Amerpet
Hyderabad
India
Amerpet
Hyderabad
India
```

Methods – Arguments and Return Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArgumentsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArgumentsExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public int Add(int a, int b)
    {
        int c;
        c = a + b;
        return (c);
    }
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        System.Console.WriteLine(s.Add(10, 20));
        System.Console.WriteLine(s.Add(50, 30));
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "C:\Users\Harsha\Google Drive\C#.NET 7.0\046. Methods\ArgumentsExample\ArgumentsExample\bin\Debug\ArgumentsExample.exe". The window contains the following text:
30
80
-

Scope of Variables

Scopes of Variables

- “Life time” of the variable is called as “scope”.
- C#.NET supports three scopes:

1. Local variables:

- Declared inside a method.
- Accessible within the same method only.

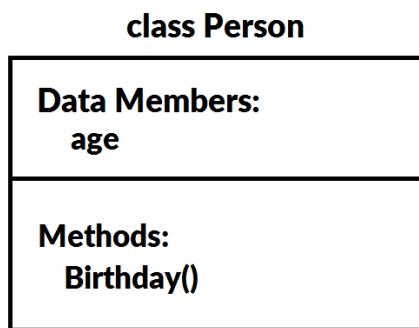
2. Block level variables:

- Declared inside a block. Ex: if block, for block etc.
- Accessible within the same block only.

3. Class level variables / Fields:

- Declared inside a class.
- Accessible within all the methods of same class.

Methods – Example - Age



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodsExample”.
- Click on OK.

Program.cs

```
class Person
{
    public int age = 20;

    public void Birthday()
    {
        age++;
    }
}
```

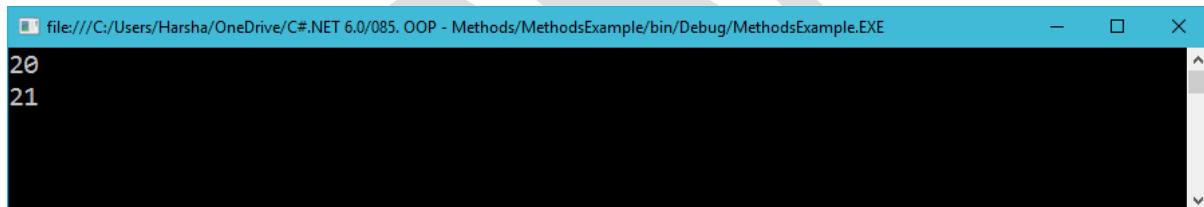
class Program

```
{  
    static void Main()  
    {  
        Person p;  
        p = new Person();  
        System.Console.WriteLine(p.age);  
        p.Birthday();  
        System.Console.WriteLine(p.age);  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/085. OOP - Methods/MethodsExample/bin/Debug/MethodsExample.EXE". The window contains the following text:
20
21

Methods – Example 2 - Numbers

```
class "Class1"
```

Data Members:

```
double a  
double b  
double sum  
double difference  
double product  
double quotient
```

Methods:

```
void Add()  
void Subtract()  
void Multiply()  
void Divide()
```

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodsExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodsExample2”.
- Click on OK.

Program.cs

```
class Class1  
{  
    public double a, b;  
    public double sum;
```

```
public double difference;
public double product;
public double quotient;

public void Add()
{
    sum = a + b;
}
public void Subtract()
{
    difference = a - b;
}
public void Multiply()
{
    product = a * b;
}
public void Divide()
{
    quotient = a / b;
}

class Program
{
    static void Main()
    {
        Class1 c1;
        c1 = new Class1();
        System.Console.Write("Enter first number : ");
        c1.a = System.Convert.ToDouble(System.Console.ReadLine());
        System.Console.Write("Enter second number : ");
        c1.b = System.Convert.ToDouble(System.Console.ReadLine());
        c1.Add();
        System.Console.WriteLine("Sum: " + c1.sum);
        c1.Subtract();
        System.Console.WriteLine("Difference: " + c1.difference);
        c1.Multiply();
        System.Console.WriteLine("Product: " + c1.product);
        c1.Divide();
        System.Console.WriteLine("Quotient: " + c1.quotient);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/086. OOP - Methods - Numbers/MethodsExample2/bin/Debug/MethodsExample2.EXE
Enter first number : 10
Enter second number : 3
Sum: 13
Difference: 7
Product: 30
Quotient: 3.333333333333333
```

Methods – Example 3 - Login

class "User"

Data Members:

Username
Password
Message

Methods:

CheckLogin()

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “LoginExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LoginExample”. Click on OK.

Program.cs

```
class User
{
    public string Username;
    public string Password;
    public string Message;
    public void CheckLogin()
    {
        if (Username == "admin" && Password == "manager")
        {
            Message = "Successful login";
        }
    }
}
```

```

    }
    else
    {
        Message = "Invalid login";
    }
}

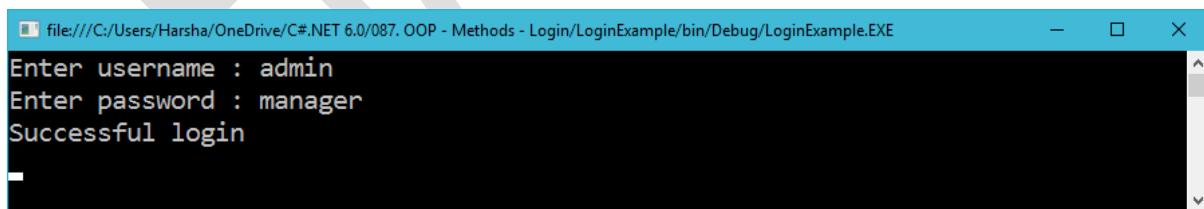
class Program
{
    static void Main()
    {
        User u;
        u = new User();
        System.Console.WriteLine("Enter username : ");
        u.Username = System.Console.ReadLine();
        System.Console.WriteLine("Enter password : ");
        u.Password = System.Console.ReadLine();
        u.CheckLogin();
        System.Console.WriteLine(u.Message);
        System.Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/087. OOP - Methods - Login/LoginExample/bin/Debug/LoginExample.EXE

```

Enter username : admin
Enter password : manager
Successful login
-
```

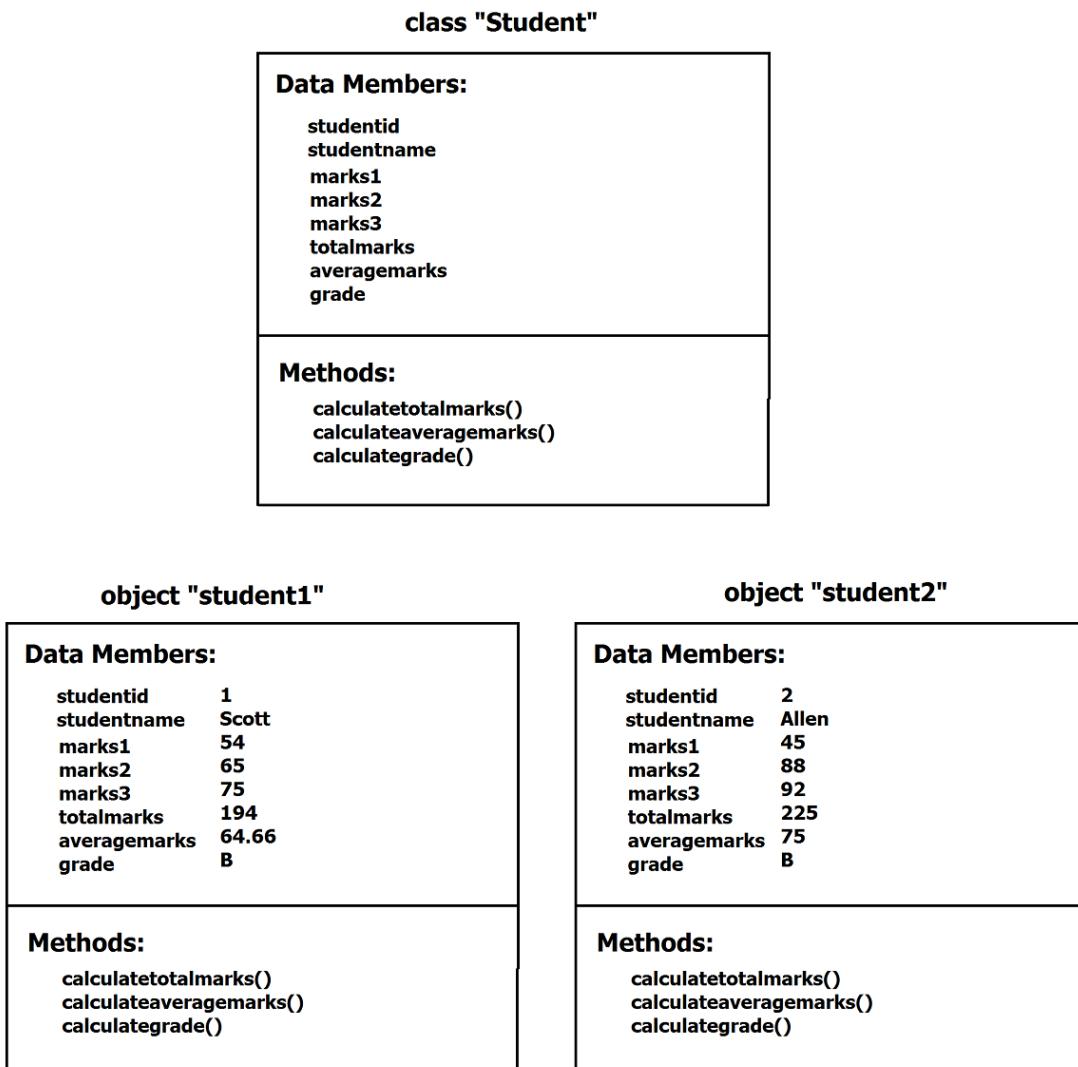


file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/087. OOP - Methods - Login/LoginExample/bin/Debug/LoginExample.EXE

```

Enter username : abc
Enter password : def
Invalid login
-
```

Methods – Example 4 - Student



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StudentExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “StudentExample”.
- Click on OK.

Program.cs

```
class Student
{
    public int studentid;
    public string studentname;
    public double marks1, marks2, marks3, totalmarks, averagemarks;
    public string grade;
    public void calculatetotalmarks()
    {
        totalmarks = marks1 + marks2 + marks3;
    }
    public void calculateaveragemarks()
    {
        averagemarks = totalmarks / 3;
    }
    public void calculategrade()
    {
        if (marks1 < 35 || marks2 < 35 || marks3 < 35)
        {
            grade = "Fail";
        }
        else if (averagemarks >= 80 && averagemarks <= 100)
        {
            grade = "A grade";
        }
        else if (averagemarks >= 60 && averagemarks < 79)
        {
            grade = "B grade";
        }
        else if (averagemarks >= 50 && averagemarks < 59)
        {
            grade = "C grade";
        }
        else if (averagemarks >= 35 && averagemarks < 49)
        {
            grade = "D grade";
        }
    }
}
```

```

        }
    }
}

class Program
{
    static void Main()
    {
        Student s1, s2;
        s1 = new Student();
        s2 = new Student();
        System.Console.Write("Enter first student id: ");
        s1.studentid = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter first student name: ");
        s1.studentname = System.Console.ReadLine();
        System.Console.Write("Enter first student marks 1: ");
        s1.marks1 = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter first student marks 2: ");
        s1.marks2 = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter first student marks 3: ");
        s1.marks3 = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.WriteLine();

        System.Console.Write("Enter second student id: ");
        s2.studentid = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter second student name: ");
        s2.studentname = System.Console.ReadLine();
        System.Console.Write("Enter second student marks 1: ");
        s2.marks1 = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter second student marks 2: ");
        s2.marks2 = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.Write("Enter second student marks 3: ");
        s2.marks3 = System.Convert.ToInt32(System.Console.ReadLine());
        System.Console.WriteLine();

        s1.calculatetotalmarks();
        s1.calculateaveragemarks();
        s1.calculategrade();

        s2.calculatetotalmarks();
        s2.calculateaveragemarks();
        s2.calculategrade();

        System.Console.WriteLine("First Student ID: " + s1.studentid);
    }
}

```

```
System.Console.WriteLine("First Student Name: " + s1.studentname);
System.Console.WriteLine("First Student Marks 1: " + s1.marks1);
System.Console.WriteLine("First Student Marks 2: " + s1.marks2);
System.Console.WriteLine("First Student Marks 3: " + s1.marks3);
System.Console.WriteLine("First Student Total Marks: " +
s1.totalmarks);
System.Console.WriteLine("First Student Average Marks: " +
s1.averagemarks);
System.Console.WriteLine("First Student Grade: " + s1.grade);
System.Console.WriteLine("-----");

System.Console.WriteLine("Second Student ID: " + s2.studentid);
System.Console.WriteLine("Second Student Name: " +
s2.studentname);
System.Console.WriteLine("Second Student Marks 1: " + s2.marks1);
System.Console.WriteLine("Second Student Marks 2: " + s2.marks2);
System.Console.WriteLine("Second Student Marks 3: " + s2.marks3);
System.Console.WriteLine("Second Student Total Marks: " +
s2.totalmarks);
System.Console.WriteLine("Second Student Average Marks: " +
s2.averagemarks);
System.Console.WriteLine("Second Student Grade: " + s2.grade);
System.Console.WriteLine("-----");

System.Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

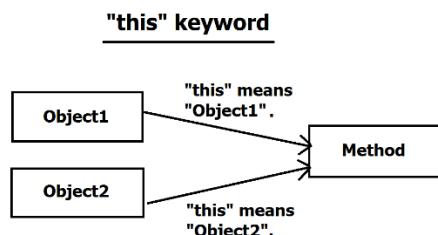
Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/088. OOP - Methods - Student/StudentExample/bin/Debug/... - X ^  
Enter first student id: 101  
Enter first student name: Scott  
Enter first student marks 1: 65  
Enter first student marks 2: 75  
Enter first student marks 3: 84  
  
Enter second student id: 102  
Enter second student name: Allen  
Enter second student marks 1: 95  
Enter second student marks 2: 45  
Enter second student marks 3: 77  
  
First Student ID: 101  
First Student Name: Scott  
First Student Marks 1: 65  
First Student Marks 2: 75  
First Student Marks 3: 84  
First Student Total Marks: 224  
First Student Average Marks: 74.66666666666667  
First Student Grade: B grade  
-----  
Second Student ID: 102  
Second Student Name: Allen  
Second Student Marks 1: 95  
Second Student Marks 2: 45  
Second Student Marks 3: 77  
Second Student Total Marks: 217  
Second Student Average Marks: 72.33333333333333  
Second Student Grade: B grade  
-----
```

"this" keyword

this keyword

- “this” is a keyword, which represents “current object”, based on which, the method or constructor was called.
- “this” keyword can be used in non-static methods and non-static constructors only. “this” keyword can’t be used in static methods or static constructors.
- It is used to access the members of current object.
- By default, the usage of “this” keyword is optional. By default system referrs to “current object”. But when a local variable name and Field name are same, then by default the system refers to local variable. Then if you want to access Field name, you must use “this.Field”. So in that way “this” keyword is must to use.



"this" keyword - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ThisExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “ThisExample”.
- Click on OK.

Program.cs

```
class Class1
{
    public int n = 10;
    public void Display()
    {
        int n = 20;
        System.Console.WriteLine("data member: " + this.n); //Output: 20
        System.Console.WriteLine("local variable: " + n); //Output: 20
        System.Console.WriteLine();
    }
}

class Program
{
    static void Main()
    {
        Class1 c1;
        c1 = new Class1();
        c1.Display();
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
C:\Users\Harsha\Google Drive\C#.NET 7.0\052.1. OOP - This\ThisExample\bin\Debug\ThisExample.exe
data member: 10
local variable: 20
```

Static Methods

Static Methods

- Non-static methods are used to manipulate non-static Fields; Static methods are used to manipulate static Fields only. Static methods can be called with class name only.
- Methods are two types:
 - Non-static methods (or) instance methods
 - Static methods (or) shared methods

Sl. No	Non-Static Methods	Static Methods
1	By default, all the methods are non-static methods. These are not created with a keyword called "static".	Static methods are created using "static" keyword.
2	Non-static methods can access both non-static members and non-static members.	Static methods can access only static members. Static methods can't access non-static members. However, if you create an object for the class in the static methods, then you can access any non-static members through the object, in the static method.
3	Non-static methods are accessible with object only. Non-static methods are not accessible without creating an object for the class.	Static methods are accessible with class only. Static methods are accessible without creating an object for the class. Static methods are not accessible with an object.
4	We can use "this" keyword in the non-static methods, because non-static methods are called with an object.	We can't use "this" keyword in the static methods, because static methods are called with class only, without an object.
5	Use non-static methods, if you want to perform some operation based on non-static Fields.	Use static Fields if you want to perform some operation based on static Fields.

Static Methods - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “StaticMethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StaticMethodsExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public static int n = 10;
    public static void Increment()
    {
        n++;
    }
}

class Program
{
    static void Main()
    {
        System.Console.WriteLine(Sample.n); //Output: 10
        Sample.Increment();
        System.Console.WriteLine(Sample.n); //Output: 11
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
G:\Users\Harsha\Google Drive\C#.NET 7.0\052.2. OOP - Static Methods\StaticMethodsExample\bin\Debug\StaticMethodsExample.exe
10
11
```

Reference Variables as Arguments

Reference Variables as Arguments

- We can pass objects as arguments from one method to another method. Then the reference (address) of the source object will be sent to the argument variable.

Reference Variables as Arguments - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReferenceVariablesAsArguments”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ReferenceVariablesAsArguments”.
- Click on OK.

Program.cs

```
class Person
{
    public int age;
}

class Sample
{
    public void Birthday(Person p)
    {
        p.age++;
    }
}
```

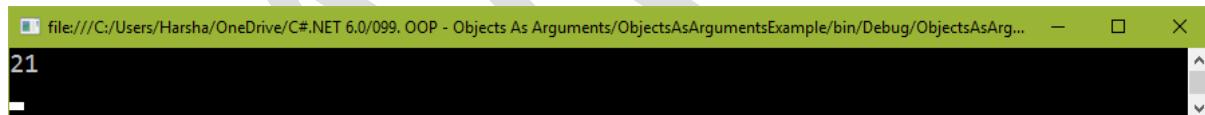
```
}
```

```
class Program
{
    static void Main()
    {
        Person p;
        p = new Person();
        Sample s;
        s = new Sample();
        p.age = 20;
        s.Birthday(p);
        System.Console.WriteLine(p.age);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/099. OOP - Objects As Arguments/ObjectsAsArgumentsExample/bin/Debug/ObjectsAsArg...  —  □  X
21
```

Reference Variables as Fields

Reference Variables as Fields

- We can create an object of one class (source class) and store its reference in a Field of another class (destination class).
- If you do so, we can access the source class's object, through the Field of destination class. Then the data type of the Field should be “source class” type.
- For example, assume you have a class called “Hyderabad”. You can create an object for “Hyderabad” class and store its reference in a Field called “h” in another class called “India”.

Reference Variables as Fields - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReferenceVariablesAsFields”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ReferenceVariablesAsFields”.
- Click on OK.

Program.cs

```
class Hyderabad
{
    public string Charminar = "Hyderabad.Charminar";
    public string Golconda = "Hyderabad.Golconda";
```

```
}
```

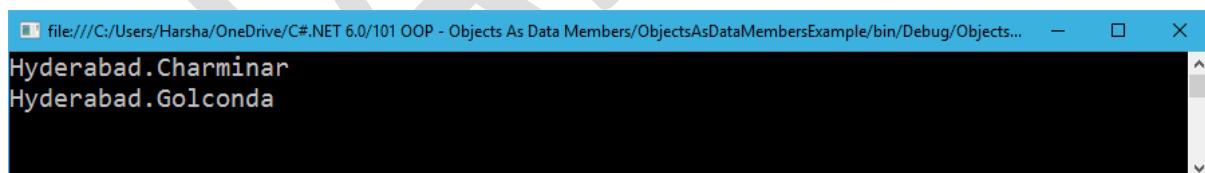
```
class India
{
    public Hyderabad h = new Hyderabad();
}

class Program
{
    static void Main()
    {
        India i;
        i = new India();
        System.Console.WriteLine(i.h.Charminar);
        System.Console.WriteLine(i.h.Golconda);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Default Arguments

Default Arguments (or) Optional Parameters

- “Default arguments” concept is used to specify a “default value” for a “argument”.
- While calling the method, if you don’t supply a value for the parameter, then the default value will be assigned to the parameter automatically.
- If you supply a value for the parameter, then the given value will be assigned to the parameter, as usual.

Syntax of method with Default Arguments

```
AccessModifier Qualifier MethodName(DataType1 ArgumentVariable1=  
DefaultValue1, DataType2 ArgumentVariable2= DefaultValue2, ...)  
{  
    Your Code here  
}
```

Default Arguments - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “DefaultArgumentsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DefaultArgumentsExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public void Display(int n = 10)
    {
        System.Console.WriteLine("n value is " + n);
    }
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        s.Display(50); //Output: 50
        s.Display(100); //Output: 100
        s.Display(); //Output: 10
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window titled "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/69. Console - Default Arguments/DefaultArgumentsExample/bin/Debug/DefaultArgu...". The window contains the following text:
n value is 50
n value is 100
n value is 10

Named Parameters

- “Named parameters” concept allows the programmer to pass a value, based on the “parameter name”, instead of depending on the order of arguments, while calling a method.
- Advantage: We can change the order of parameters.
- Syntax:

MethodName(ParameterName: value, ParameterName: value, ...)

Named Parameters - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “NamedParametersExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NamedParametersExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public void Display(int x, int y)
    {
```

```
System.Console.WriteLine("x is: " + x);
System.Console.WriteLine("y is: " + y);
}

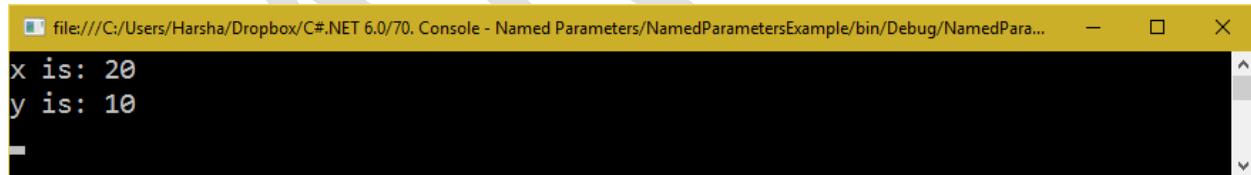
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        s.Display(y: 10, x: 20); //Output: x is 20, y is 10
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/70. Console - Named Parameters/NamedParametersExample/bin/Debug/NamedPara...". The main area of the window displays the following text:
x is: 20
y is: 10

Methods Overloading

- “Method Overloading” concept is “writing multiple methods with same name within the same class, with different types of arguments”.
- The difference can be in data types of arguments or in no. of arguments.
- When you call a method, it calls the method, which matches with the given arguments.
- Example:
 - ◆ Method1()
 - ◆ Method1(int a)
 - ◆ Method1(double d)
 - ◆ Method1(int a, double d)
 - ◆ Method1(double d, int a)
 - ◆ Method1(string s)
 - etc.

Methods Overloading - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodOverloadingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodOverloadingExample”.
- Click on OK.

Program.cs

```

class Sample
{
    //method 1
    public void Display(int n)
    {
        System.Console.WriteLine("int: " + n);
    }

    //method 2
    public void Display(string s)
    {
        System.Console.WriteLine("string: " + s);
    }
}

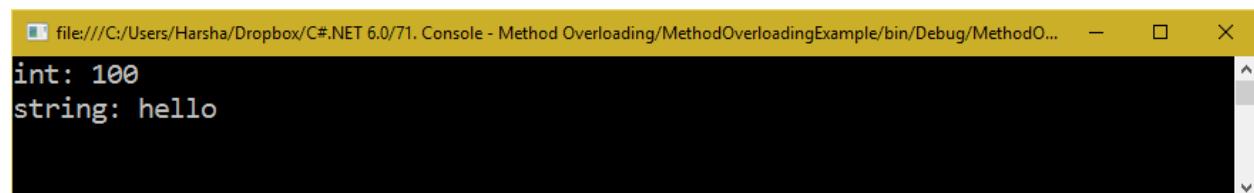
class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        int a = 100;
        string b = "hello";
        s.Display(a); //calls method1
        s.Display(b); //calls method2
        System.Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/71. Console - Method Overloading/MethodOverloadingExample/bin/Debug/MethodO...
int: 100
string: hello

```

Types of Parameters

Types of Parameters

1. Call by value
2. Call by reference
3. Call by output

Call by Value

- The changes made to the “argument variable” will not be effected in “original variable”.
- This is default in c#.net.

Call by Reference

- The changes made to the “argument variable” will be effected automatically in the “original variable”.
- This is implemented using “ref” keyword with “original variable” and “argument variable”.
- The “original variable” should be a variable (can’t be a literal).

Call by Output

- This is same as “call by reference”.
- The difference between “call by reference” and “call by output” is: the value of “original variable” will not be transferred to the “argument variable”; but the value comes back from “argument variable” to “original variable”, at the end of method.
- The “argument variable” will be “unassigned” by default.
- The “argument variable” must be set to a value before the method definition ends.
- This is implemented using “out” keyword in both calling portion and receiving portion.
- The “original variable” should be a variable; can’t be a literal.

Call by value - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “CallByValueExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CallByValueExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public void Method1(int x)
    {
        System.Console.WriteLine(x); //Output: 100
        x = 150;
        System.Console.WriteLine(x); //Output: 150
    }
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        int a = 100;
        s.Method1(a);
        System.Console.WriteLine(a); //Output: 100
    }
}
```

```
    System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/72. Console - Call by value/CallByValueExample/bin/Debug/CallByValueExample.EXE  
100  
150  
100
```

Call by reference - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CallByReferenceExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CallByReferenceExample”.
- Click on OK.

Program.cs

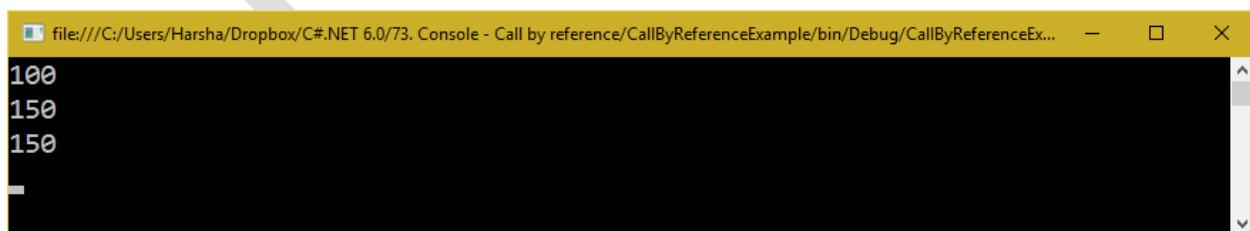
```
class Sample
{
    public void Method1(ref int x)
    {
        System.Console.WriteLine(x); //Output: 100
        x = 150;
        System.Console.WriteLine(x); //Output: 150
    }
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        int a = 100;
        s.Method1(ref a);
        System.Console.WriteLine(a); //Output: 150
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
100
150
150
```

Call by output - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CallByOutExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CallByOutExample”.
- Click on OK.

Program.cs

```
class Sample
{
    public void Method1(out int x)
    {
        x = 150;
        System.Console.WriteLine(x); //Output: 150
    }
}

class Program
{
    static void Main()
    {
        Sample s;
        s = new Sample();
        int a;
        s.Method1(out a);
        System.Console.WriteLine(a); //Output: 150
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/74. Console - Call by output/CallByOutExample/bin/Debug/CallByOutExample.EXE
150
150
```

Type Conversion

Type Conversion in C#.NET

- The process of converting a value from “one data type” to “another data type” is called as “type conversion”.
- C# supports 4 types of type conversion:
 1. Implicit Casting
 2. Explicit Casting (or) Type Casting
 3. Parsing
 4. Conversion Methods

Implicit Casting

- C# compiler automatically converts a value from “lower numeric data type” to “higher numeric data type”. This automatic conversion is called “implicit casting”.
- We will compare the no. of data types of the data type, to identify whether it is a lower data type / higher data type.
- Ex: “int” to “long”. Here “int” is 4 bytes and “long” is 8 bytes. So, it is “lower” to “higher”.
- As implicit casting will be performed automatically, there is no syntax for this. Directly you can assign a value of “lower numeric data type” to “higher numeric data type”.

Implicit Casting - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ImplicitCastingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ImplicitCastingExample”. Click on OK.

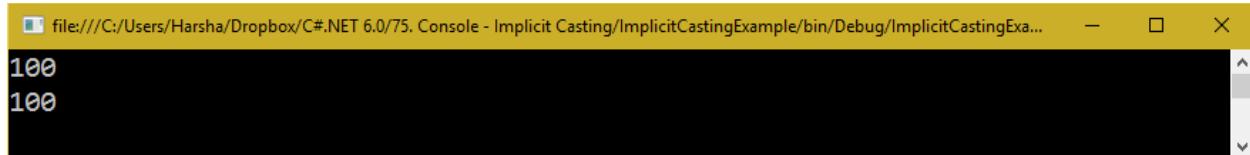
Program.cs

```
class Program
{
    static void Main()
    {
        //A variable of "lower numerical data type" i.e. "short" (2 bytes)
        short a = 100;
        //A variable of "higher numerical data type" i.e. "int" (4 bytes)
        int b;
        //short to int = lower to higher = implicit casting
        b = a;
        System.Console.WriteLine(a); //Output: 100
        System.Console.WriteLine(b); //Output: 100
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/75. Console - Implicit Casting/ImplicitCastingExample/bin/Debug/ImplicitCastingExa...". The window contains the following text:
100
100

Explicit Casting (or) Type Casting

- It is used to convert a value from “lower numerical data type” to “higher numerical data type” and also from “higher numerical data type” to “lower numerical data type”.
- Syntax: *(Expected data type) value*

Explicit Casting (or) Type Casting - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “TypeCastingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TypeCastingExample”.
- Click on OK.

Program.cs

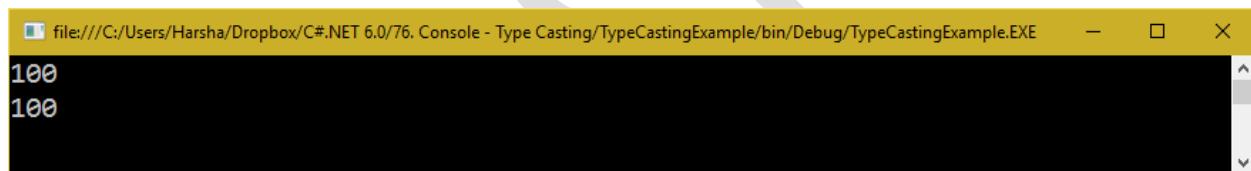
```
class Program
{
    static void Main()
    {
        //create variable of "int" data type
        int a = 100;
        //create variable of "short" data type
        short b;
```

```
//convert the value from "int" data type "short" data type (higher to  
lower), using "type casting" concept  
b = (short)a;  
System.Console.WriteLine(a); //Output: 100  
System.Console.WriteLine(b); //Output: 100  
System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Parsing

- It is used to convert a value from “string” data type to “any numerical data type”.
- “Parse” is a pre-defined method. Parsing means converting.
- Syntax: *NumericalDataTypeName.Parse(value);*
- Rule: The string value should contain digits only; otherwise it can't be converted into numerical data type.
- Note: If the string value is not convertible into “numerical data type”, you will get exception (run time error).

Parsing - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ParsingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ParsingExample”.
- Click on OK.

Program.cs

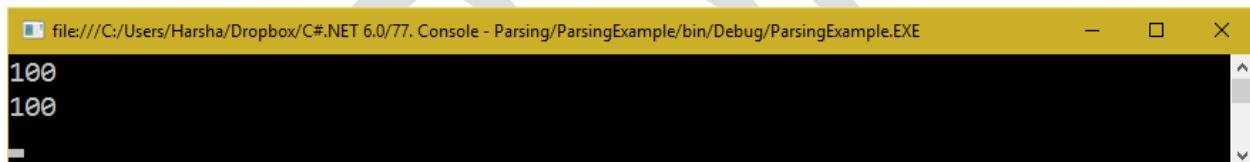
```
class Program
{
    static void Main()
    {
        //create variable of "string" data type
        string s = "100";
    }
}
```

```
//create variable of "int" data type
int x;
//convert the value from "string" to "int" data type, using Parsing
concept
x = int.Parse(s);
System.Console.WriteLine(s); //Output: 100
System.Console.WriteLine(x); //Output: 100
System.Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/77. Console - Parsing/ParsingExample/bin/Debug/ParsingExample.EXE". The window contains the following text:
100
100
-

TryParse

- When you are converting "alphabets" or "alpha-numeric value" from "string" data type to "numeric", you will get a run time error (exception).
- "TryParse" is a pre-defined method, which is used to avoid while converting the value from "string" to "numerical data type".
- It tries to convert the string to numerical data type. If conversion is successful, it returns "true". If conversion is failed, it returns "false". If conversion is successful, it stores the result value in the destination variable; otherwise, it stores 0 (zero) in the destination variable.
- Syntax: `bool result = NumericalDataTypeName.TryParse(source string value, out destination numerical variable);`

TryParse - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “TryParseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TryParseExample”.
- Click on OK.

Program.cs

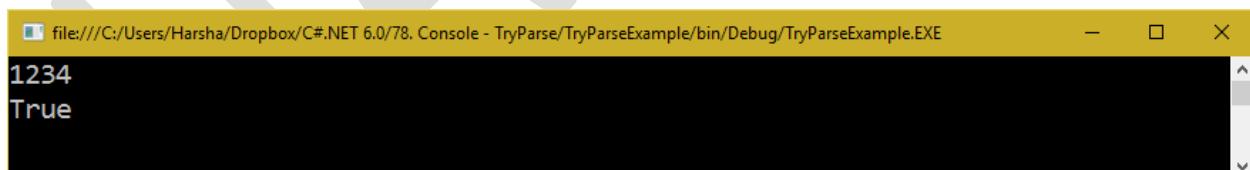
```
class Program
{
```

```
static void Main()
{
    //create string & integer
    string s = "1234";
    int n;
    //try to convert "1234" into "int" data type. If the conversion is
    successful, it returns the value into the "out" parameter & "true" directly. If
    the conversion is unsuccessful, it returns "false".
    bool b = int.TryParse(s, out n);
    //display
    System.Console.WriteLine(n); //Output: 1234
    System.Console.WriteLine(b); //Output: True
    System.Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/78. Console - TryParse/TryParseExample/bin/Debug/TryParseExample.EXE
1234
True
```

TryParse – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “TryParseExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TryParseExample2”.
- Click on OK.

Program.cs

```
class Program
{
    static void Main()
    {
        //create string and integer
        string s = "hyderabad1234";
        int n;

        //try to convert "hyderabad1234" into int data type. Actually it not
        //possible to convert "alphanumeric value" like "hyderabad1234" into "int"
        //data type. So it returns "0" into "n" and "false" into "b".
        bool b = int.TryParse(s, out n);
        System.Console.WriteLine(n); //Output: 0
        System.Console.WriteLine(b); //Output: False
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/79. Console - TryParse 2/TryParseExample2/bin/Debug/TryParseExample2.EXE
0
False
-
```

Conversion Methods

Conversion Methods

- C#.NET provides a set of conversion methods, which are used to convert a value from “any standard data type” to “any other standard data type”.
- Standard data types all numerical and non-numerical data types.
- Syntax: `System.Convert.MethodName(value);`

List of conversion methods		
Sl. No	Conversion Method	Description
1	<code>byte System.Convert.ToByte(value)</code>	It converts and returns the given value into “byte” data type.
2	<code>sbyte System.Convert.ToSByte(value)</code>	It converts and returns the given value into “sbyte” data type.
3	<code>short System.Convert.ToInt16(value)</code>	It converts and returns the given value into “short” data type.
4	<code>ushort System.Convert.ToUInt16(value)</code>	It converts and returns the given value into “ushort” data type.
5	<code>int System.Convert.ToInt32(value)</code>	It converts and returns the given value into “int” data type.
6	<code>uint System.Convert.ToUInt32(value)</code>	It converts and returns the given value into “uint” data type.
7	<code>long System.Convert.ToInt64(value)</code>	It converts and returns the given value into “long” data type.
8	<code>ulong System.Convert.ToUInt64(value)</code>	It converts and returns the given value into “ulong” data type.

9	<code>float System.Convert.ToSingle(value)</code>	It converts and returns the given value into “float” data type.
10	<code>double System.Convert.ToDouble(value)</code>	It converts and returns the given value into “double” data type.
11	<code>decimal System.Convert.ToDecimal(value)</code>	It converts and returns the given value into “decimal” data type.
12	<code>char System.Convert.ToChar(value)</code>	It converts and returns the given value into “char” data type.
13	<code>string System.Convert.ToString(value)</code>	It converts and returns the given value into “string” data type.
14	<code>bool System.Convert.ToBoolean (value)</code>	It converts and returns the given value into “bool” data type.
15	<code>DateTime System.Convert.ToDateTime(value)</code>	It converts and returns the given value into “DateTime” data type.

- Note: You have to use the conversion method, based on the destination data type (target data type), into which you want to convert.

Conversion Methods - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “ConversionMethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ConversionMethodsExample”.
- Click on OK.

Program.cs

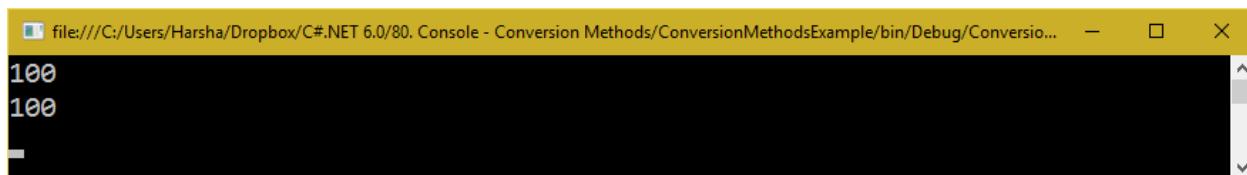
```
class Program
{
    static void Main()
    {
        //create a variable of "double" data type
        double d = 100;
        //create a variable of "int" data type
        int n;
        //convert the value from "double" data type to "int" data type", using
        conversion method
        n = System.Convert.ToInt32(d);
        System.Console.WriteLine(d); //Output: 100
        System.Console.WriteLine(n); //Output: 100
        System.Console.ReadKey();
    }
}

//Note: Try all the remaining conversion methods.
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/Dropbox/C#.NET 6.0/80. Console - Conversion Methods/ConversionMethodsExample/bin/Debug/ConversionMethodsExample". The window contains the following text:
100
100
-
The window has standard minimize, maximize, and close buttons at the top right.

Constructors

- Constructors are special methods of the class, which initializes the Fields and perform any initialization process, which will be called automatically when an object is created for the class.
- Constructors are used to receive a set of parameters and initialize those parameters into respective Fields.
- For example, you have a class with Fields. So your constructor can receive 3 arguments and assign those arguments into Fields respectively. However, if you don't want to initialize all the Fields, you can receive less no. of arguments also. For example, you can initialize 1 or 2 Fields also.
- When an object is created, the following process happens internally:
 - Memory will be allocated for the object in RAM (in Heap of RAM).
 - Default values will be initialized (based on the data type).
 - Constructor will be called automatically.

Syntax of Constructor

```
public Classname(datatype variable1, datatype variable2, ...)  
{  
    Field1 = variable1;  
    Field2 = variable2;  
    ...  
}
```

Types of Constructor

- Parameter-less constructor: A constructor that has no arguments. It initializes Fields with some fixed values.
- Parameterized constructor: A constructor that has one or more arguments. It initializes Fields with argument (varied) values.

Default Constructor

- As per the rules of c#.net, a class should have a constructor. If you don't create a constructor in the class, .net creates an empty constructor (implicit constructor or default constructor) automatically and internally.

Parameterless Constructor - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ParameterlessConstructorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ParameterlessConstructorExample”.
- Click on OK.

Program.cs

```
class Sample
{
    //Fields
    public int a, b;

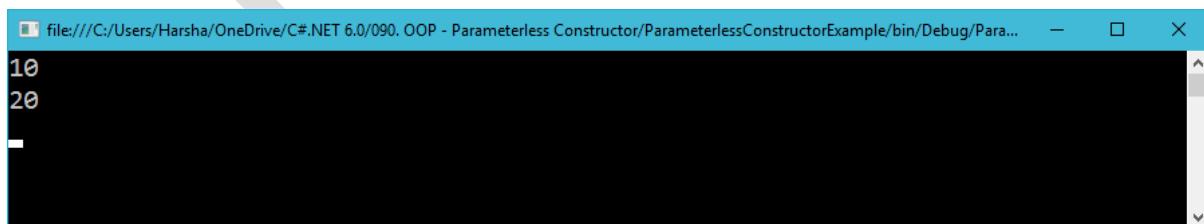
    //constructor
    public Sample()
```

```
{  
    a = 10;  
    b = 20;  
}  
}  
  
class Program  
{  
    static void Main()  
    {  
        Sample s;  
  
        s = new Sample();  
  
        System.Console.WriteLine(s.a);  
        System.Console.WriteLine(s.b);  
  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a terminal window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/090. OOP - Parameterless Constructor/ParameterlessConstructorExample/bin/Debug/Para...". The window displays the following text:
10
20
-
The window has standard operating system controls (minimize, maximize, close) at the top right.

Parameterized Constructor - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ParameterizedConstructorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ParameterizedConstructorExample”.
- Click on OK.

Program.cs

```
class Sample
{
    //Fields
    public int a, b;

    //constructor
    public Sample(int x, int y)
    {
        a = x;
        b = y;
    }
}

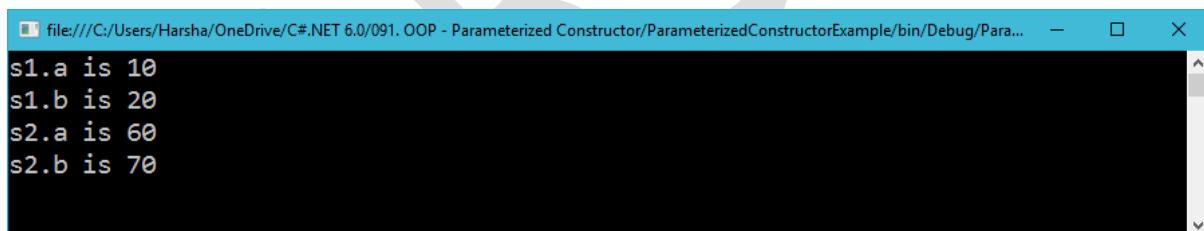
class Program
{
    static void Main()
    {
        //creating reference variable
```

```
Sample s1, s2;  
//creating objects  
s1 = new Sample(10, 20);  
s2 = new Sample(60, 70);  
  
//get data  
System.Console.WriteLine("s1.a is " + s1.a);  
System.Console.WriteLine("s1.b is " + s1.b);  
System.Console.WriteLine("s2.a is " + s2.a);  
System.Console.WriteLine("s2.b is " + s2.b);  
System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
s1.a is 10  
s1.b is 20  
s2.a is 60  
s2.b is 70
```

Constructors - Student - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ConstructorsStudentExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ConstructorsStudentExample”.
- Click on OK.

Program.cs

```
//create a class called "Student" with 3 Fields called "studentid",
"studentname" and "marks"
class Student
{
    //Fields
    public int studentid;
    public string studentname;
    public int marks;

    //constructor (to receive parameters & initialize them into respective
    Fields)
    public Student(int a, string b, int c)
    {
        studentid = a;
        studentname = b;
        marks = c;
    }
}
```

```

class Program
{
    static void Main()
    {
        //create 3 reference variables of "Student" type
        Student s1, s2, s3;
        //create 3 objects of "Student" type
        s1 = new Student(101, "Scott", 70);
        s2 = new Student(102, "Allen", 80);
        s3 = new Student(103, "John", 90);
        //get data from first object
        System.Console.WriteLine("First student:");
        System.Console.WriteLine("Student ID: " + s1.studentid);
        System.Console.WriteLine("Student Name: " + s1.studentname);
        System.Console.WriteLine("Marks: " + s1.marks);
        System.Console.WriteLine("-----");
        //get data from second object
        System.Console.WriteLine("Second student:");
        System.Console.WriteLine("Student ID: " + s2.studentid);
        System.Console.WriteLine("Student Name: " + s2.studentname);
        System.Console.WriteLine("Marks: " + s2.marks);
        System.Console.WriteLine("-----");
        //get data from third object
        System.Console.WriteLine("Third student:");
        System.Console.WriteLine("Student ID: " + s3.studentid);
        System.Console.WriteLine("Student Name: " + s3.studentname);
        System.Console.WriteLine("Marks: " + s3.marks);
        System.Console.WriteLine("-----");
        System.Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/092. OOP - Constructors - Student/ConstructorsStudentExample/bin/Debug/Constructors... - X
```

```
First student:  
Student ID: 101  
Student Name: Scott  
Marks: 70  
-----  
Second student:  
Student ID: 102  
Student Name: Allen  
Marks: 80  
-----  
Third student:  
Student ID: 103  
Student Name: John  
Marks: 90  
-----
```

Object Initializer

Object Initializer

- Drawback of constructor: Once we have created a constructor with some set of arguments, every time we call it, you must pass the same no. of values; we can't initialize desired Fields; so there is no flexibility in the constructor.
- Object initializer solves the above problem.
- "Object initializer" is a syntax, which is used to initialize "desired" Fields.
- "Object initializer" concept introduced in c#.net 3.0.
- "Object initializer" will be called after constructor.
- Syntax of creating object based on a class using Object Initializer:

```
{ Field1 = Value1, Field2 = Value2, ... };
```

- When an object is created, the following process happens internally:
 - First, necessary memory will be allocated for the object in RAM.
 - Default values will be initialized into the Fields.
 - Next, constructor will be called automatically.
 - Next, object initializer will be called automatically.

Object Initializer - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ObjectInitializerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ObjectInitializerExample”.
- Click on OK.

Program.cs

```
//create a class called "Student" with 3 Fields called "studentid",
"studentname" and "marks"
class Student
{
    //Fields
    public int studentid;
    public string studentname;
    public int marks;
}

class Program
{
    static void Main()
    {
        //create 3 reference variables of "Student" type
        Student s1, s2, s3;

        //create 3 objects of "Student" type
        s1 = new Student() { studentid = 101, studentname = "Scott", marks = 70 };
        s2 = new Student() { studentid = 102, studentname = "Allen", marks = 80 };
        s3 = new Student() { studentid = 103, studentname = "John", marks = 90 };

        //get data from first object
        System.Console.WriteLine("First student:");
        System.Console.WriteLine("Student ID: " + s1.studentid);
```

```
System.Console.WriteLine("Student Name: " + s1.studentname);
System.Console.WriteLine("Marks: " + s1.marks);
System.Console.WriteLine("-----");
//get data from second object
System.Console.WriteLine("Second student:");
System.Console.WriteLine("Student ID: " + s2.studentid);
System.Console.WriteLine("Student Name: " + s2.studentname);
System.Console.WriteLine("Marks: " + s2.marks);
System.Console.WriteLine("-----");
//get data from third object
System.Console.WriteLine("Third student:");
System.Console.WriteLine("Student ID: " + s3.studentid);
System.Console.WriteLine("Student Name: " + s3.studentname);
System.Console.WriteLine("Marks: " + s3.marks);
System.Console.WriteLine("-----");
System.Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/093. OOP - Object Initializer/ObjectInitializerExample/bin/Debug/ObjectInitializerExample.E... - X
```

```
First student:  
Student ID: 101  
Student Name: Scott  
Marks: 70  
-----  
Second student:  
Student ID: 102  
Student Name: Allen  
Marks: 80  
-----  
Third student:  
Student ID: 103  
Student Name: John  
Marks: 90  
-----
```

"Set" and "Get" Methods

- “Set methods” and “get methods” are used to perform “validation” on a private Field.
- You must create the Field as “private Field”, for security.

Set Method

- Set method receives a value as argument, checks (validates) it, and then assigns the same into the Field.
- Set method protects the Field from invalid values.
- Set method acts as mediator between Main method and Field.
- Set method is used to perform validations and makes the program stronger and systematic.
- Note: When set method is created, it is recommended to make the Field as “private”. The private members are accessible only within the same class.
- Syntax of Set Method:

```
public void SetFieldname(Datatype Argument)
{
    Field= Argument;
}
```

Get Method

- Get method returns the value of a Field.
- Get method can perform automatic calculations.
- Syntax of Get Method:

```
public Returntype GetFieldname()
{
    return (Field);
}
```

"Set" and "Get" Methods - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SetAndGetMethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SetAndGetMethodsExample”.
- Click on OK.

Program.cs

```
//create a class called "Person"
class Person
{
    //Field
    public int age;

    //set method
    public void SetAge(int value)
    {
        if (value >= 18 && value <= 60)
        {
            age = value;
        }
    }

    //get method
    public int GetAge()
    {
```

```
    return (age);
}
}

class Program
{
    static void Main()
    {
        Person p = new Person();

        p.SetAge(70);
        System.Console.WriteLine(p.GetAge());

        p.SetAge(24);
        System.Console.WriteLine(p.GetAge());

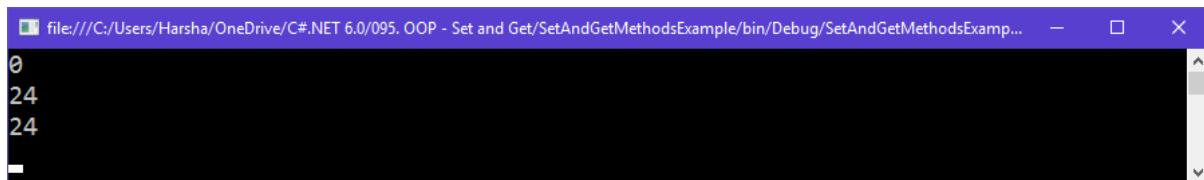
        p.SetAge(240);
        System.Console.WriteLine(p.GetAge());

        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/095. OOP - Set and Get/SetAndGetMethodsExample/bin/Debug/SetAndGetMethodsExamp...
0
24
24
```

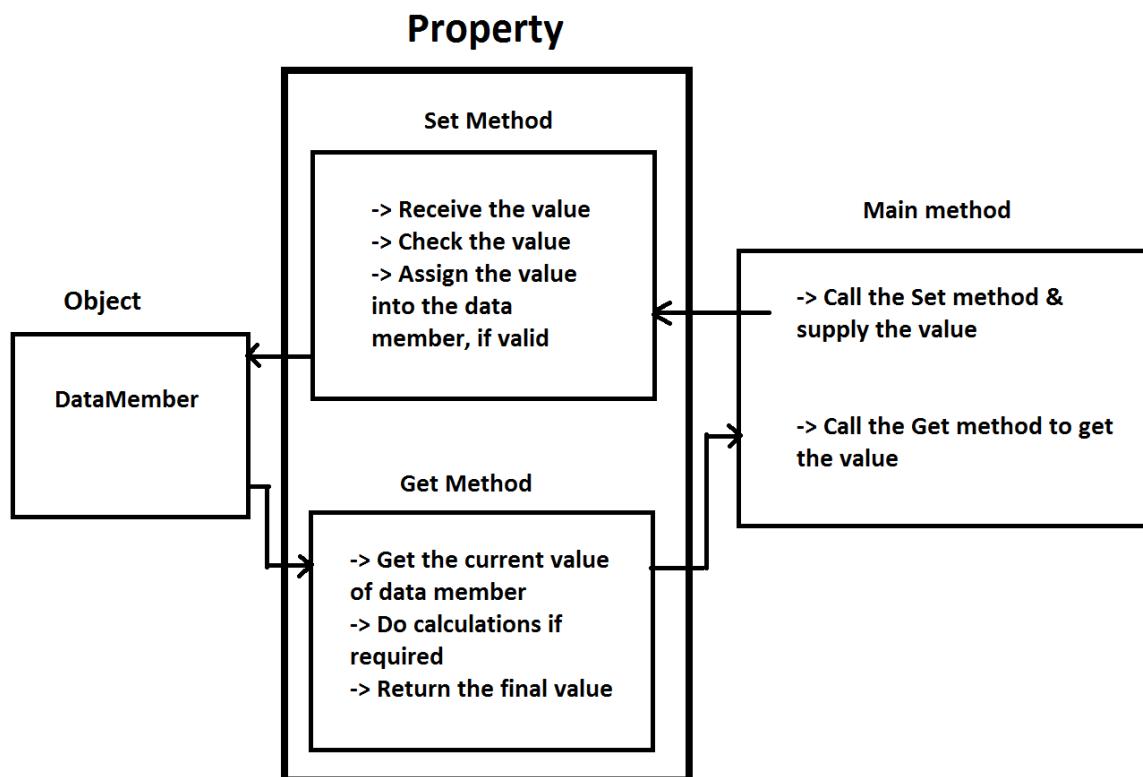
Properties

- A property is a group of “set method” and “get method”, which perform validation on a private Field.
- Properties are used to implement a protection layer around Field.
- Properties can't store any value.
- Properties are used for validations and automatic calculations. The “set method” is used to create validations; and “get method” is used to create automatic calculations or return the value of a Field as it is.
- Properties offer simple way to call “set method” and “get method”.
- In real time, for every private Field, we create a property. That means the private Field is not accessible directly in the client code; though property it can be accessible indirectly.
- The property's set method will be called automatically when a value is assigned to a property.
- The property's set method has an implicit argument called “value” that represents the value assigned to the property.
- The property's get method will be called automatically when the client code gets the value of the property.
- The property's get method has no arguments but should return the value.
- Note: While creating properties, it is recommended to create Field name start with “_” (underscore). The underscore has no meaning, but it indicates that it is Field.

Syntax to create a property:

```
public Datatype Propertyname
{
    set
    {
        Field= value;
    }
}
```

```
get
{
    return (Field);
}
```



Properties - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “PropertiesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “PropertiesExample”.
- Click on OK.

Program.cs

```
//create a class called "Person"
class Person
{
    //private Field
    private int _age;

    //property (set + get)
    public int Age
    {
        set
        {
            if (value >= 18 && value <= 60)
            {
                _age = value;
            }
        }
        get
        {
```

```
        return _age;
    }
}

}

class Program
{
    static void Main()
    {
        Person p = new Person();
        p.Age = 70;
        System.Console.WriteLine(p.Age);
        p.Age = 24;
        System.Console.WriteLine(p.Age);
        p.Age = 240;
        System.Console.WriteLine(p.Age);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a terminal window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/096. OOP - Properties/PropertiesExample/bin/Debug/PropertiesExample.EXE". The window shows the following text:
0
24
24

Readonly and WriteOnly Properties

Read-only properties

- Read-only properties contain only get method; no set method.
- The user can get the value of read-only property, but can't set the value.
- Syntax to create a readonly property:

```
public Datatype Propertynname  
{  
    get  
    {  
        return (Field);  
    }  
}
```

Write-only properties

- Write-only properties contain only set method; no get method.
- The user can set the value of write-only property, but can't get the value.
- Syntax to create a write-only property:

```
public Datatype Propertynname  
{  
    set  
    {  
        Field = value;  
    }  
}
```

Readonly Properties - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReadOnlyPropertiesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ReadOnlyPropertiesExample”.
- Click on OK.

Program.cs

```
class Student
{
    //private Fields
    public int _marks;
    public string _result;
    //properties
    public int Marks
    {
        set
        {
            if (value >= 0 && value <= 100)
            {
                _marks = value;
            }
        }
        get
        {
            return (_marks);
```

```

        }
    }

//readonly property (only get method)
public string Result
{
    get
    {
        if (_marks < 35)
            _result = "Fail";
        else
            _result = "Pass";
        return _result;
    }
}
class Program
{
    static void Main()
    {
        Student s = new Student();
        s.Marks = 70;
        System.Console.WriteLine(s.Result);
        System.Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Automatic Properties

Automatic properties

- Automatic properties are created using { get; set; } syntax.
- When you create automatic property, the compiler automatically creates the following things automatically at compilation time:
 - Private Field
 - Set method
 - Get method
- Advantage of automatic properties: Easy syntax to create simple properties.
- But automatic properties are useful only when you don't want to perform any validation. If you want to perform validation, use normal properties.
- Syntax:

```
public Datatype Propertname{ get; set; }
```

Automatic Properties - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.5”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AutomaticPropertiesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AutomaticPropertiesExample”.

- Click on OK.

Program.cs

```
class Person
{
    //automatic properties
    public string PersonName { get; set; }
    public int Age { get; set; }
}

class Program
{
    static void Main()
    {
        Person p = new Person();
        p.PersonName = "Scott";
        p.Age = 22;
        System.Console.WriteLine(p.PersonName);
        System.Console.WriteLine(p.Age);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/098. OOP - Automatic Properties/AutomaticPropertiesExample/bin/Debug/AutomaticProp...". The window contains the following text:
Scott
22

Inheritance

What is Inheritance

- It is a OOP concept of creating “parent-child” relationship among two or more classes.
- It is a concept of extending an existing class, by creating another class based on the existing class.
- Here, the classes are two types: parent class and child class.
- Child class extends the parent class.
- Parent class also called as “base class” or “super class”.
- Child class also called as “derived class” or “sub class”.
- As a result of inheritance, all the members of parent class will become as members of child class automatically.
- When an object of child class is created, it creates an object of parent class internally, and it will be part of the child class’s object.
- One parent class can have one or more child classes; but one child class can have ONLY ONE parent class.
- If you create a new child class based on an existing child class, all its members and also its parent class members will be inherited into the new child class.
- Syntax of inheritance:

```
class Childclassname: Parentclassname
{
}
```

Inheritance - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InheritanceExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InheritanceExample”.
- Click on OK.

Program.cs

```
//parent class
class Class1
{
    public int a = 10;
    public int b = 20;
}

//child class
class Class2 : Class1
{
    public int c = 30;
    public int d = 40;
}

class Program
{
    static void Main()
    {
        //create a reference variable for child class (Class2)
        Class2 p;

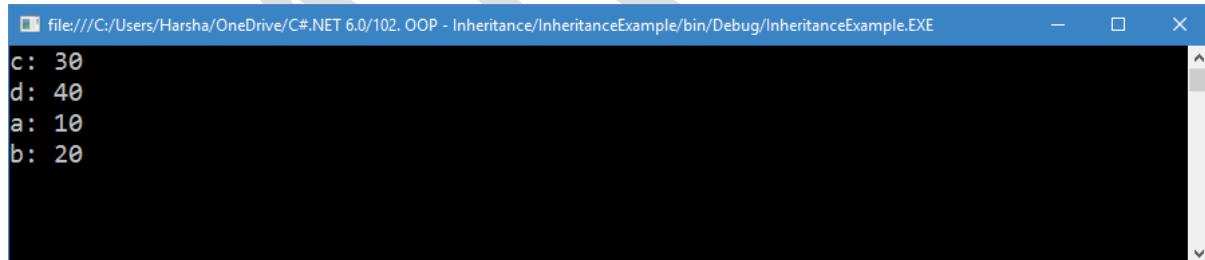
        //create object for child class (Class2); It internally creates an object for
        //parent class also. That means it allocates memory for "a", "b", "c" and "d".
        p = new Class2();
```

```
//child class Fields  
System.Console.WriteLine("c: " + p.c);  
System.Console.WriteLine("d: " + p.d);  
  
//parent class Fields  
System.Console.WriteLine("a: " + p.a);  
System.Console.WriteLine("b: " + p.b);  
  
System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
c: 30  
d: 40  
a: 10  
b: 20
```

Inheritance – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InheritanceExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InheritanceExample2”.
- Click on OK.

Program.cs

```
//parent class
class Person
{
    public string PersonName;
    public string Gender;
}

//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;
}

//child class 2
class Employee : Person
{
    public int EmpID;
    public double Salary;
```

```
}
```

```
class Program
{
    static void Main()
    {
        ***** creating object for parent class *****
        Person p;
        p = new Person();
        p.PersonName = "Scott";
        p.Gender = "Male";
        System.Console.WriteLine("Person Name: " + p.PersonName);
        System.Console.WriteLine("Gender: " + p.Gender);
        System.Console.WriteLine("\n");

        ***** creating object for child class 1 *****
        Student s;
        s = new Student();
        s.PersonName = "Allen";
        s.Gender = "Male";
        s.StudentID = 101;
        s.Marks = 70;
        System.Console.WriteLine("Student Name: " + s.PersonName);
        System.Console.WriteLine("Gender: " + s.Gender);
        System.Console.WriteLine("Student ID: " + s.StudentID);
        System.Console.WriteLine("Student Marks: " + s.Marks);
        System.Console.WriteLine("\n");

        ***** creating object for child class 2 *****
        Employee emp;
        emp = new Employee();
        emp.PersonName = "Jones";
        emp.Gender = "Male";
        emp.EmpID = 201;
        emp.Salary = 7000;
        System.Console.WriteLine("Employee Name: " + emp.PersonName);
        System.Console.WriteLine("Gender: " + emp.Gender);
```

```
System.Console.WriteLine("Employee ID: " + emp.EmpID);
System.Console.WriteLine("Employee Salary: " + emp.Salary);

System.Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/103. OOP - Inheritance 2/InheritanceExample/bin/Debug/InheritanceExample.EXE
Person Name: Scott
Gender: Male

Student Name: Allen
Gender: Male
Student ID: 101
Student Marks: 70

Employee Name: Jones
Gender: Male
Employee ID: 201
Employee Salary: 7000
```

“base” keyword

Calling the parent class's member from child class's method:

- To access the members of parent class within the child class's method:
 - Syntax: `base.parentclassmembername`
- Note: Here, “base” is a keyword that represents the “object of parent class”, which is a part of “object of child class”.

“base” keyword - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “BaseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “BaseExample”.
- Click on OK.

Program.cs

```
//parent class
class Person
{
    public string PersonName;
    public string Gender;
}
```

```

//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;

    //Student.DisplayStudentDetails
    public void DisplayStudentDetails()
    {
        System.Console.WriteLine("Student Name: " + base.PersonName);
        System.Console.WriteLine("Gender: " + base.Gender);
        System.Console.WriteLine("Student ID: " + this.StudentID);
        System.Console.WriteLine("Student Marks: " + this.Marks);
    }
}

//child class 2
class Employee : Person
{
    public int EmpID;
    public double Salary;

    //Employee.DisplayEmployeeDetails
    public void DisplayEmployeeDetails()
    {
        System.Console.WriteLine("Employee Name: " + base.PersonName);
        System.Console.WriteLine("Gender: " + base.Gender);
        System.Console.WriteLine("Employee ID: " + this.EmpID);
        System.Console.WriteLine("Employee Salary: " + this.Salary);
    }
}

class Program
{
    static void Main()
    {
        /***** creating object for child class 1 *****/
    }
}

```

```

Student s;
s = new Student();
s.PersonName = "Allen";
s.Gender = "Male";
s.StudentID = 101;
s.Marks = 70;
s.DisplayStudentDetails();
System.Console.WriteLine("\n");

***** creating object for child class 2 *****
Employee emp;
emp = new Employee();
emp.PersonName = "Jones";
emp.Gender = "Male";
emp.EmpID = 201;
emp.Salary = 7000;
emp.DisplayEmployeeDetails();

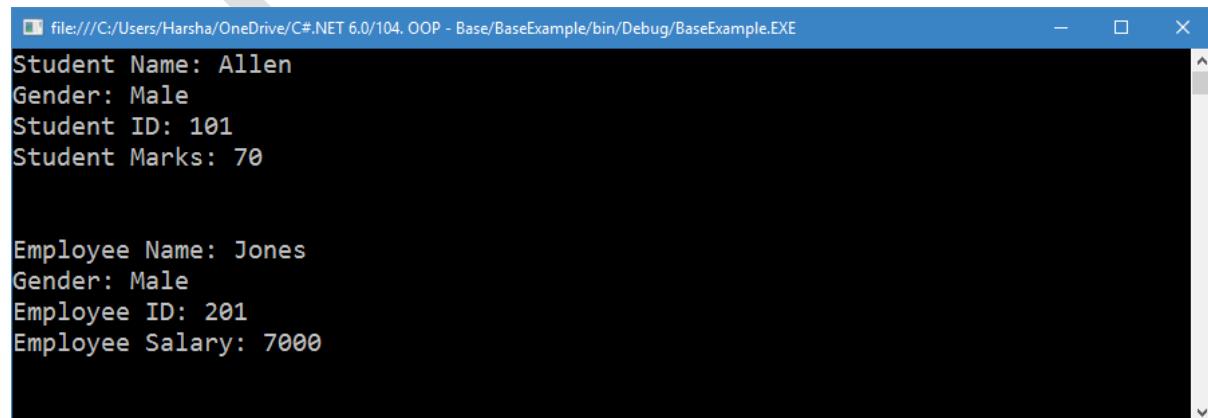
System.Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/104. OOP - Base/BaseExample/bin/Debug/BaseExample.EXE
Student Name: Allen
Gender: Male
Student ID: 101
Student Marks: 70

Employee Name: Jones
Gender: Male
Employee ID: 201
Employee Salary: 7000

```

Parent class's constructor

Calling the parent class's constructor from child class's constructor:

- The parent class's parameterless constructor will be called automatically, when child class's constructor was called. But you must call the parent class's parameterized constructor explicitly in the child class's constructor.
- In the child class's constructor, you can use the following syntax to call the parent class's parameterized constructor:
 - `base(parameter1, parameter2, ...)`

○ `base(parameter1, parameter2, ...)`

Parent class's constructor - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ParentClassConstructorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ParentClassConstructorExample”.
- Click on OK.

Program.cs

```
//parent class
class Person
{
    public string PersonName;
    public string Gender;

    //parent class constructor
    public Person(string PersonName, string Gender)
    {
        this.PersonName = PersonName;
        this.Gender = Gender;
    }
}

//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;

    //child class's constructor
    //We are receiving 4 values; First two values will be forwarded to the
    parent class constructor; Remaining two values will be stored in the child
    class's object.
    public Student(string PersonName, string Gender, int StudentID, int
    Marks) : base(PersonName, Gender)
    {
        this.StudentID = StudentID;
        this.Marks = Marks;
    }

    //Student.DisplayStudentDetails
    public void DisplayStudentDetails()
    {
        System.Console.WriteLine("Student Name: " + base.PersonName);
        System.Console.WriteLine("Gender: " + base.Gender);
    }
}
```

```

System.Console.WriteLine("Student ID: " + this.StudentID);
System.Console.WriteLine("Student Marks: " + this.Marks);
}

}

//child class 2
class Employee : Person
{
    public int EmpID;
    public double Salary;

    //child class's constructor
    //We are receiving 4 values; First two values will be forwarded to the
    parent class constructor; Remaining two values will be stored in the child
    class's object.
    public Employee(string PersonName, string Gender, int EmpID, int
Salary) : base(PersonName, Gender)
    {
        this.EmpID = EmpID;
        this.Salary = Salary;
    }

    //Employee.DisplayEmployeeDetails
    public void DisplayEmployeeDetails()
    {
        System.Console.WriteLine("Employee Name: " + base.PersonName);
        System.Console.WriteLine("Gender: " + base.Gender);
        System.Console.WriteLine("Employee ID: " + this.EmpID);
        System.Console.WriteLine("Employee Salary: " + this.Salary);
    }
}

class Program
{
    static void Main()
    {
        /***** creating object for child class 1 *****/
    }
}

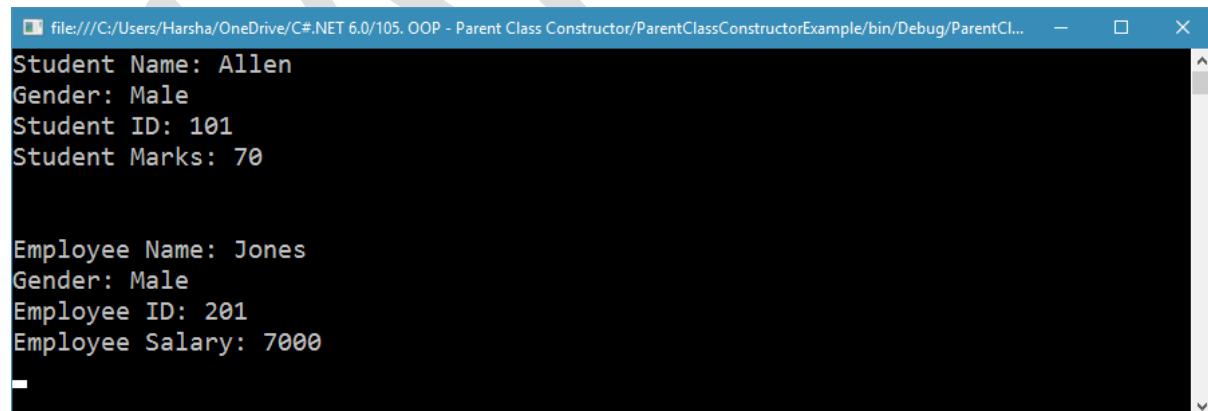
```

```
Student s;  
s = new Student("Allen", "Male", 101, 70);  
s.DisplayStudentDetails();  
System.Console.WriteLine("\n");  
  
/****** creating object for child class 2 *****/  
Employee emp;  
emp = new Employee("Jones", "Male", 201, 7000);  
emp.DisplayEmployeeDetails();  
  
System.Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/105. OOP - Parent Class Constructor/ParentClassConstructorExample/bin/Debug/ParentCl... - X  
Student Name: Allen  
Gender: Male  
Student ID: 101  
Student Marks: 70  
  
Employee Name: Jones  
Gender: Male  
Employee ID: 201  
Employee Salary: 7000
```

Method Hiding

Method Hiding

- “Method hiding” is used to hide the parent class’s method, by creating another method in the child class, with same signature (access modifier, method name and arguments).
- For method hiding, we need not use any special keyword in the parent class’s method. But we have to use “new” keyword in the child class’s method optionally. The “new” keyword specifies the compiler that we are doing the “method hiding” intentionally.

Syntax to create normal method (in parent class):

```
public Returntype Methodname(Datatype1 Argument1, DataType2 Argument2, ...)
{
    Code here
}
```

Syntax to create hiding method (in child class):

```
public new Returntype Methodname(Datatype1 Argument1, DataType2 Argument2, ...)
{
    Code here
}
```

Method Hiding - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodHidingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodHidingExample”.
- Click on OK.

Program.cs

```
//parent class
class Class1
{
    //Class1.Display
    public void Display()
    {
        System.Console.WriteLine("Class1.Display");
    }
}

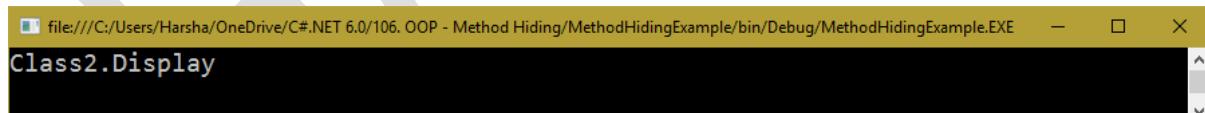
//child class
class Class2 : Class1
{
    //Class2.Display
    //Method hiding: This method hides the parent class's method called
    "Display". Both methods are having the same signature; but code is
    different. Note: The "new" keyword is optional here.
    public new void Display()
```

```
{  
    System.Console.WriteLine("Class2.Display");  
}  
}  
  
class Program  
{  
    static void Main()  
    {  
        Class2 c2;  
        c2 = new Class2();  
        c2.Display(); //calls Class2.Display only  
  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Method Hiding – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodHidingExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodHidingExample2”.
- Click on OK.

Program.cs

```
//parent class
class Person
{
    //Fields of "Person" class
    public string PersonName;
    public string Gender;

    //Person.Display
    public void Display()
    {
        System.Console.WriteLine("Person Name: " + PersonName);
        System.Console.WriteLine("Gender: " + Gender);
    }
}

//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;
```

```
//Student.Display  
//Method Hiding: This method hides the parent class's method called  
"Display". Both methods are having the same signature; but code is  
different.  
public new void Display()  
{  
    System.Console.WriteLine("Student Name: " + PersonName);  
    System.Console.WriteLine("Gender: " + Gender);  
    System.Console.WriteLine("Student ID: " + StudentID);  
    System.Console.WriteLine("Student Marks: " + Marks);  
}  
}  
//child class 2  
class Employee : Person  
{  
    public int EmpID;  
    public double Salary;  
  
    //Employee.Display  
    //Method Hiding: This method hides the parent class's method called  
"Display". Both methods are having the same signature; but code is  
different.  
    public new void Display()  
    {  
        System.Console.WriteLine("Employee Name: " + PersonName);  
        System.Console.WriteLine("Gender: " + Gender);  
        System.Console.WriteLine("Employee ID: " + EmpID);  
        System.Console.WriteLine("Employee Salary: " + Salary);  
    }  
}  
class Program  
{  
    static void Main()  
    {  
        /***** creating object for child class 1 *****/  
        Student s;
```

```

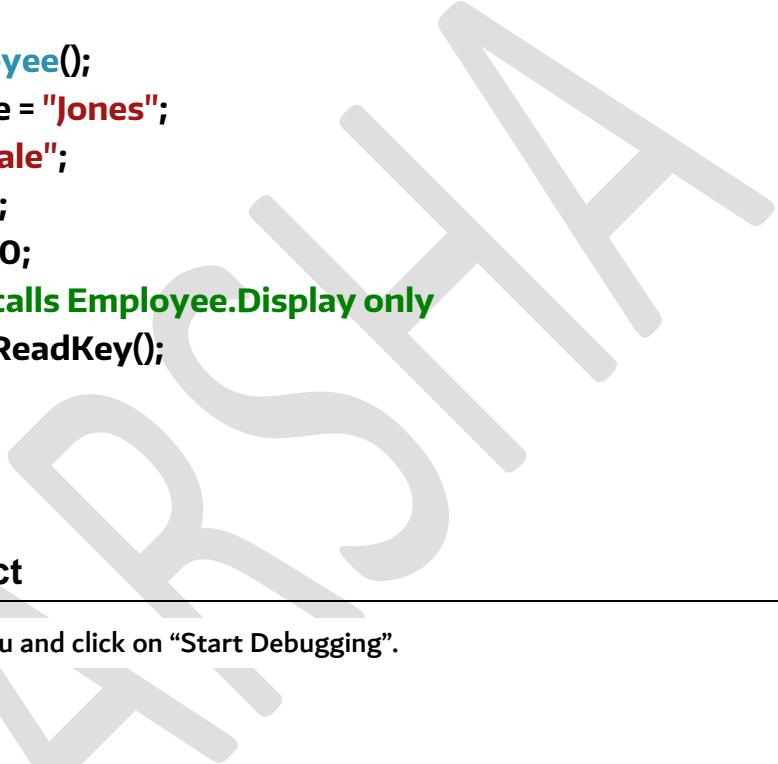
s = new Student();
s.PersonName = "Allen";
s.Gender = "Male";
s.StudentID = 101;
s.Marks = 70;
s.Display(); //calls Student.Display only
System.Console.WriteLine();
***** creating object for child class 2 *****/
Employee emp;
emp = new Employee();
emp.PersonName = "Jones";
emp.Gender = "Male";
emp.EmpID = 201;
emp.Salary = 7000;
emp.Display(); //calls Employee.Display only
System.Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window showing the output of the C# program. The output displays two sets of data: one for a Student object and one for an Employee object, both using their respective Display() methods.

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/107. OOP - Method Hiding 2/MethodHidingExample/bin/Debug/MethodHidingExample.E...
Student Name: Allen
Gender: Male
Student ID: 101
Student Marks: 70

Employee Name: Jones
Gender: Male
Employee ID: 201
Employee Salary: 7000
-
```

Method Overriding

Virtual Methods and Method Overriding

- “Method Overriding” is a concept of “extending a parent class’s method”, by creating another method in the child class, with same name and same signature.
- The parent class’s method should be “virtual method”. The child class’s method should be “override method”.
- The “virtual” keyword in parent class’s method allows the child classes to override (redefine) the parent class’s method in the child classes.
- The “override” keyword in child class’s method informs the parent class that it is overriding the parent class’s virtual method.
- The virtual methods only can be overridden.
- The method signature (method name, access modifier, arguments, and return type) should be same in between virtual method and override method.
- Virtual methods can’t be private.
- Overriding the virtual methods is optional.
- In real time, we use “method overriding” to extend an existing method, which is created by another programmer.

Normal Methods (vs) Virtual Methods

Sl. No	Normal Method	Virtual Method
1	No "virtual" keyword	Should have "virtual" keyword.
2	Present in any class (parent class or in child class).	Should be present in parent class only.

Method Hiding (vs) Method Overriding

- "Method Hiding" is similar to "Method Overriding", but having the following differences.

Sl. No	Method Hiding	Method Overriding
1	We need not use any keyword in parent class method; "new" keyword in child class method.	We should use "virtual" keyword in parent class method; "override" keyword in child class method.
2	Method hiding is used to hide (overwrite) the parent class's method. That means parent class's method will not execute; only child class's method executes.	Method overriding is used to extend the parent class's method. That means both parent class's method and child class's method execute.

Syntax to create virtual method (in parent class):

```
public virtual Returntype Methodname(Datatype1 Argument1,  
Datatype2 Argument2, ...)  
{  
    Code here  
}
```

Syntax to create override method (in child class):

```
public override Returntype Methodname(Datatype1 Argument1,  
Datatype2 Argument2, ...)  
{  
    Code here  
}
```

Method Overriding - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodOverridingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodOverridingExample”.
- Click on OK.

Program.cs

```
//parent class
class Class1
{
    //Class1.Display (virtual method)
    public virtual void Display()
    {
        System.Console.WriteLine("Class1.Display");
    }
}

//child class
class Class2 : Class1
{
    //Class2.Display
    //Method overriding: This method overrides (extends) the parent class's
    //virtual method called "Display". That means both methods will execute.
    //Both methods should have same signature.
    public override void Display()
```

```
{  
    base.Display();  
    System.Console.WriteLine("Class2.Display");  
}  
}  
  
class Program  
{  
    static void Main()  
    {  
        Class2 myobject;  
        myobject = new Class2();  
        //call the both methods Class1.Display and Class2.Display  
        myobject.Display();  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/108. OOP - Method Overriding/MethodOverridingExample/bin/Debug/MethodOverriding...  
Class1.Display  
Class2.Display
```

Method Overriding – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MethodOverridingExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MethodOverridingExample2”.
- Click on OK.

Program.cs

```
//parent class
class Person
{
    public string PersonName;
    public string Gender;

    //Person.Display (virtual method)
    public virtual void Display()
    {
        System.Console.WriteLine("Person Name: " + PersonName);
        System.Console.WriteLine("Gender: " + Gender);
    }
}

//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;
```

```

//Student.Display
//Method overriding: This method overrides (extends) the parent class's
virtual method called "Display". Both methods will execute. Both methods
should have same signature.
public override void Display()
{
    base.Display();
    System.Console.WriteLine("Student ID: " + StudentID);
    System.Console.WriteLine("Student Marks: " + Marks);
}
}

//child class 2
class Employee : Person
{
    public int EmpID;
    public double Salary;
    //Employee.Display
    //Method overriding: This method overrides (extends) the parent class's
    virtual method called "Display". Both methods will execute. Both methods
    should have same signature.
    public override void Display()
    {
        base.Display();
        System.Console.WriteLine("Employee ID: " + EmpID);
        System.Console.WriteLine("Employee Salary: " + Salary);
    }
}

class Program
{
    static void Main()
    {
        //***** creating object for child class 1 *****/
        Student s;
        s = new Student();
        s.PersonName = "Allen";
        s.Gender = "Male";
    }
}

```

```

s.StudentID = 101;
s.Marks = 70;
s.Display(); //calls to Person.Display and Employee.Display
System.Console.WriteLine("\n");
***** creating object for child class 2 *****/
Employee emp;
emp = new Employee();
emp.PersonName = "Jones";
emp.Gender = "Male";
emp.EmpID = 201;
emp.Salary = 7000;
emp.Display(); //calls to Person.Display and Student.Display
System.Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/109. OOP - Method Overriding 2/MethodOverridingExample2/MethodOverridingExample2...
Person Name: Allen
Gender: Male
Student ID: 101
Student Marks: 70

Person Name: Jones
Gender: Male
Employee ID: 201
Employee Salary: 7000
-
```

Abstract Classes

Abstract Classes

- Abstract classes are similar to normal classes and can contain abstract members additionally, and those can't be instantiated.
- Abstract classes are created using “abstract” keyword
- Abstract classes contain a set of members that are common to all of its child classes.
- Abstract classes can be inherited. That means we can create a child class based on the abstract class.
- We can't create object for the abstract class directly. The only way to utilize its members is “creating object for child class and calling abstract class's members through the child class's object”.
- Syntax to create an abstract class:

```
abstract class Classname
{
}
```

Normal Class (vs) Abstract Class

Sl. No	Normal Class (or) Non-abstract class	Abstract class
1	No "abstract" keyword	Should have "abstract" keyword.
2	We can create an object for normal class.	We can't create an object for abstract class; but we can create a reference variable.
3	Normal class can't contain abstract methods.	Abstract class can contain abstract methods.

Abstract Classes - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AbstractClassesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AbstractClassesExample”. Click on OK.

Program.cs

```
//parent class (abstract class)
abstract class Person
{
    //normal method in abstract class
    public void Display()
    {
        System.Console.WriteLine("Person.Display");
    }
}

//child class 1
class Student : Person
{
}

//child class 2
class Employee : Person
{
}

class Program
{
```

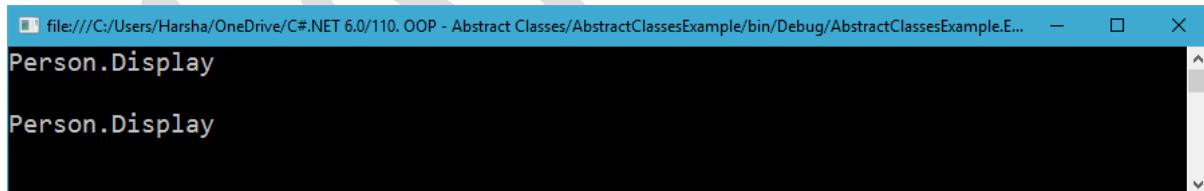
```
static void Main()
{
    //***** creating object for child class 1 *****/
    Student s;
    s = new Student();
    s.Display();
    System.Console.WriteLine("\n");

    //***** creating object for child class 1 *****/
    Employee emp;
    emp = new Employee();
    emp.Display();
    System.Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/110. OOP - Abstract Classes/AbstractClassesExample/bin/Debug/AbstractClassesExample.E...
Person.Display
Person.Display
```

Abstract Classes – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AbstractClassesExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AbstractClassesExample2”.
- Click on OK.

Program.cs

```
//parent class (abstract class)
abstract class Person
{
    public string PersonName;
    public string Gender;

    //normal method in abstract class
    public void Display()
    {
        System.Console.WriteLine("Person name: " + this.PersonName);
        System.Console.WriteLine("Gender: " + this.Gender);
    }
}

//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;
```

```
}

//child class 2
class Employee : Person
{
    public int EmpID;
    public double Salary;
}

class Program
{
    static void Main()
    {
        //***** creating object for child class 1 *****/
        Student s;
        s = new Student();
        s.PersonName = "Allen";
        s.Gender = "Male";
        s.StudentID = 101;
        s.Marks = 70;
        s.Display();
        System.Console.WriteLine("Student ID: " + s.StudentID);
        System.Console.WriteLine("Marks: " + s.Marks);
        System.Console.WriteLine("\n");
        //***** creating object for child class 2 *****/
        Employee emp;
        emp = new Employee();
        emp.PersonName = "Jones";
        emp.Gender = "Male";
        emp.EmpID = 201;
        emp.Salary = 7000;
        emp.Display();
        System.Console.WriteLine("Employee ID: " + emp.EmpID);
        System.Console.WriteLine("Salary: " + emp.Salary);
        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/111. OOP - Abstract Classes 2/AbstractClassesExample/bin/Debug/AbstractClassesExample...
Person name: Allen
Gender: Male
Student ID: 101
Marks: 70

Person name: Jones
Gender: Male
Employee ID: 201
Salary: 7000
```

Abstract Methods

Abstract Methods

- Abstract methods are “declaration-only” method that are declared in the abstract class (parent class) and must be implemented (overridden) in the child class(es).
- Abstract methods can be created only within the abstract class.
- Abstract methods contain only declaration (signature or prototype).
- Signature = Method name, return type, arguments and access modifier.
- Abstract methods should be overridden in the child class, using “override” keyword.
- In real time, abstract class with abstract method(s) will be created by one programmer and its child class(es) will be created by another programmer.
- Assume that there are two programmers. The first programmer is creating parent class (cum abstract class); the second programmer is creating child class. But both are present in two different components (parts) of the project. So whenever the first programmer can't write the actual code for a method in the parent class and wants to let the second programmer two write actual code for the method, then he will create an abstract method in the parent class. So the first programmer writes method signature only; but not the method definition. Then the second programmer has to create a child class, based on the existing parent class. Then the second programmer must create method definition. This work is called as “method overriding with abstract methods”.
- Note: “Method overriding of virtual methods” is optional; but “method overriding of abstract methods” is must.
- Virtual methods contain method body also. Abstract methods contain signature only, no method body.

Normal Method (vs) Abstract Method

Sl. No	Normal method (or) Non-abstract method	Abstract method
1	No "abstract" keyword	Should have "abstract" keyword.
2	We can call the normal method.	We can't call the abstract method.
3	Normal methods can be created in both normal class and also in abstract class.	Abstract methods can be created in only abstract class.
4	Normal methods contain method signature and body.	Abstract methods contain only method signature; no method body.
5	Normal methods can be present in both parent class and also in the child class.	Abstract methods can be present only in the parent class.
6	Normal methods can be 'private'.	Abstract methods can't be 'private'.

Virtual Method (vs) Abstract Method

Sl. No	Virtual method	Abstract method
1	Should have "virtual" keyword.	Should have "abstract" keyword.
2	We can call the virtual method.	We can't call the abstract method.
3	Virtual methods can be created in both normal class and also in abstract class.	Abstract methods can be created in only abstract class.
4	Virtual methods contain method signature and body.	Abstract methods contain only method signature; no method body.
5	Overriding the virtual methods is optional.	Overriding the abstract methods is must.

Syntax to create abstract method (in parent class):

**public abstract Returntype Methodname(Datatype1
Argument1, DataType2 Argument2, ...);**

Syntax to create override method (in child class):

**public override Returntype Methodname(Datatype1
Argument1, DataType2 Argument2, ...)
{
 Code here
}**

Abstract Methods - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AbstractMethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AbstractMethodsExample”.
- Click on OK.

Program.cs

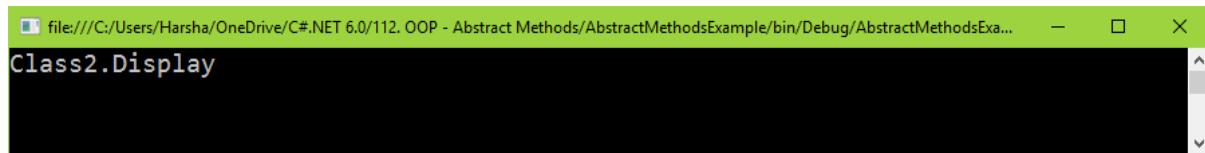
**//parent class (abstract class)
abstract class Class1**

```
{  
    //abstract method in parent class. Abstract method doesn't contain  
    method body. It contains only signature.  
    public abstract void Display();  
}  
  
//child class  
class Class2 : Class1  
{  
    //Class2.Display  
    //Method overriding: This method overrides the parent class's abstract  
    method called "Display". Both methods are having the same signature.  
    public override void Display()  
    {  
        System.Console.WriteLine("Class2.Display");  
    }  
}  
  
class Program  
{  
    static void Main()  
    {  
        Class2 c2;  
        c2 = new Class2();  
        c2.Display();  
  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Abstract Methods – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AbstractMethodsExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AbstractMethodsExample2”.
- Click on OK.

Program.cs

```
//parent class (abstract class)
abstract class Person
{
    public string PersonName;
    public string Gender;

    //abstract method in parent class. Abstract method doesn't contain
    //method body. It contains only method signature.
    public abstract void Display();
}
```

```
//child class 1
class Student : Person
{
    public int StudentID;
    public int Marks;

    //Student.Display
    //Method Overriding: This method overrides the parent class's abstract
    method called "Display". Both methods are having the same signature.
    public override void Display()
    {
        System.Console.WriteLine("Student Name: " + base.PersonName);
        System.Console.WriteLine("Gender: " + base.Gender);
        System.Console.WriteLine("Student ID: " + this.StudentID);
        System.Console.WriteLine("Student Marks: " + this.Marks);
    }
}

//child class 2
class Employee : Person
{
    public int EmplID;
    public double Salary;

    //Employee.Display
    //Method overriding: This method overrides the parent class's abstract
    method called "Display". Both methods are having the same signature.
    public override void Display()
    {
        System.Console.WriteLine("Employee Name: " + base.PersonName);
        System.Console.WriteLine("Gender: " + base.Gender);
        System.Console.WriteLine("Employee ID: " + this.EmplID);
        System.Console.WriteLine("Employee Salary: " + this.Salary);
    }
}
```

```
class Program
{
    static void Main()
    {
        //***** creating object for child class 1 *****/
        Student s;
        s = new Student();
        s.PersonName = "Allen";
        s.Gender = "Male";
        s.StudentID = 101;
        s.Marks = 70;
        s.Display();
        System.Console.WriteLine("\n");

        //***** creating object for child class 2 *****/
        Employee emp;
        emp = new Employee();
        emp.PersonName = "Jones";
        emp.Gender = "Male";
        emp.EmpID = 201;
        emp.Salary = 7000;
        emp.Display();

        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/113. OOP - Abstract Methods 2/AbstractMethodsExample/bin/Debug/AbstractMethodsExa... — X
Student Name: Allen
Gender: Male
Student ID: 101
Student Marks: 70

Employee Name: Jones
Gender: Male
Employee ID: 201
Employee Salary: 7000
-
```

Interfaces

- Interfaces are similar to abstract classes, but interfaces contain only abstract methods and property declarations.
- We can't create other members such as Fields, normal methods, constructors, properties etc., in interface.
- It is recommended to start the name of the interface with "I".
- We can't create an object for the interface, but we can create one or more child classes for interface.
- Interface provides a list of abstract methods that should be overridden in the child classes. By default, all the interface methods "public" and "abstract" automatically.
- All the child classes that inherits (implements) the interface should implement (override) all the interface methods. But different child classes can implement the interface methods with different code. If you don't implement (override) any of the interface methods in the child class, you will get compile-time error.
- Real-time use of interface: Interface acts as "contract" or "bridge" between two parts of a project. Assume that there are two programmers that create two different components of the project. The interface acts as bridge between both the programmers. This situation occurs in advanced .net frameworks such as WPF, MVC etc.
 - The first programmer creates an interface; then creates an interface variable and calls the method, without knowing its implementation.
 - The second programmer creates a child class for the interface and implements all of its methods, without knowing its calling.

Abstract Class (vs) Interface

Sl. No	Abstract Class	Interface

1	Abstract class can contain all types of members such as Fields, methods, properties, constructors etc. & also abstract methods.	Can have only abstract methods & property declarations.
2	A child class can't inherit multiple abstract classes.	A child class can inherit (implement) multiple interfaces
3	Abstract class can't act as a mediator between two classes.	Interface act as a mediator between two classes. In one class, we will implement in the interface. In the other class, we will create a reference variable for the interface & call the methods.
4	Use abstract class, if you want to provide a list of non-abstract methods and abstract methods to the child classes.	Use interface, if you want to provide only a list of abstract methods to the child classes.

5	<p>Use abstract class, if you want to create a “partial generic type”.</p> <p>Ex: “Vehicle” is an abstract class, which provides common methods to all of its child classes such as “Car”, “Bike”, and “Bus” etc.</p> <p>Here the “Vehicle” class contains both non-abstract (normal) methods and also abstract methods.</p>	<p>Use interface, if you want to create pure “generic type”.</p> <p>Ex: “IHolidayTrip” is an interface, which provides common abstract methods to all of its child classes such as “FamilyHolidayTrip”, “CollegeHolidayTrip” and “CompanyHolidayTrip”.</p> <p>If “CompanyHolidayTrip” is an abstract (parent) class, “TCSHolidayTrip”, “WiproHolidayTrip” etc., are the child classes.</p> <p>Observation:</p> <ul style="list-style-type: none"> • “IHolidayTrip” is pure general-purpose. It doesn’t matter, whether it is college holiday trip or company holiday trip. It just provides a list of common methods of all types of holiday trips. • “CompanyHolidayTrip” is partial general-purpose. It provides the list of methods that are common for all company holiday trips; but it doesn’t matter which company holiday trip it is.
---	---	---

Syntax to create an interface:

```
interface Interfacename  
{  
    Declare interface methods here  
}
```

Syntax to create a child class for the interface:

```
class Classname: Interfacename  
{  
    Implement all the interface methods here  
}
```

Interfaces - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InterfacesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InterfacesExample”.
- Click on OK.

Program.cs

```
//interface  
interface Interface1
```

```

{
    //abstract methods
    void Display();
}

//child class
class Class1 : Interface1
{
    //Method Overriding: This method overrides (implements) the
    "Interface1.Display" method. Both methods are having the same signature.
    public void Display()
    {
        System.Console.WriteLine("Class1.Display");
    }
}

class Program
{
    static void Main()
    {
        Class1 p;
        p = new Class1();
        p.Display();

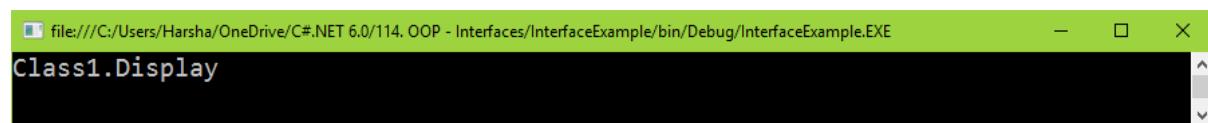
        System.Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/114. OOP - Interfaces/InterfaceExample/bin/Debug/InterfaceExample.EXE
Class1.Display
```

Interfaces – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InterfacesExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InterfacesExample2”.
- Click on OK.

Program.cs

```
//interface
interface IPerson
{
    //properties
    string PersonName { get; set; }
    string Gender { get; set; }

    //abstract method
    void Display();
}

//child class 1
class Student : IPerson
{
    //properties
    public string PersonName { get; set; }
    public string Gender { get; set; }
    public int StudentID { get; set; }
    public int Marks { get; set; }
```

```
//implementing the IPerson.Display method
public void Display()
{
    System.Console.WriteLine("Student Name: " + PersonName);
    System.Console.WriteLine("Gender: " + Gender);
    System.Console.WriteLine("Student ID: " + StudentID);
    System.Console.WriteLine("Student Marks: " + Marks);
}
}

//child class 2
class Employee : IPerson
{
    //properties
    public string PersonName { get; set; }
    public string Gender { get; set; }
    public int EmpID { get; set; }
    public double Salary { get; set; }

    //implementing the IPerson.Display method
    public void Display()
    {
        System.Console.WriteLine("Employee Name: " + PersonName);
        System.Console.WriteLine("Gender: " + Gender);
        System.Console.WriteLine("Employee ID: " + EmpID);
        System.Console.WriteLine("Employee Salary: " + Salary);
    }
}

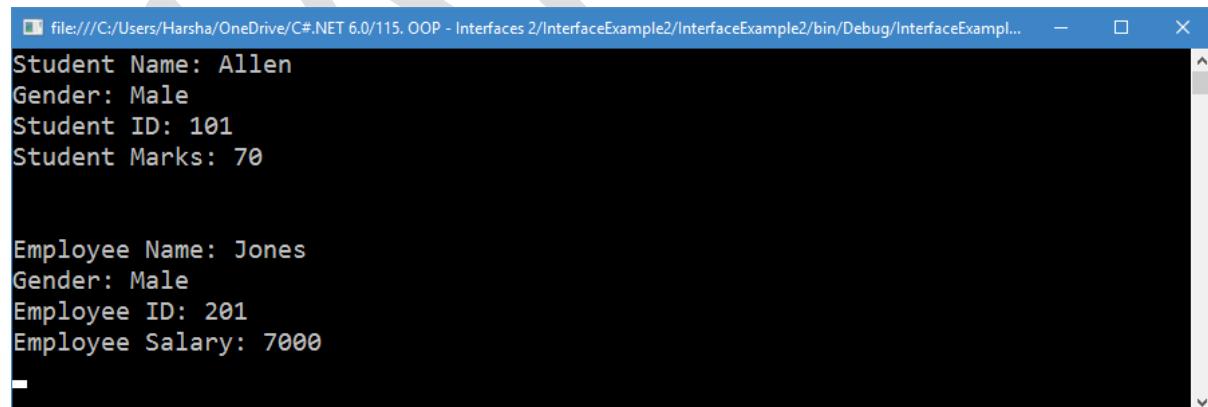
class Program
{
    static void Main()
    {
        //***** creating object for child class 1 *****/
        Student s;
        s = new Student();
        s.PersonName = "Allen";
        s.Gender = "Male";
        s.StudentID = 101;
```

```
s.Marks = 70;
s.Display();
System.Console.WriteLine("\n");
/********** creating object for child class 2 *****/
Employee emp;
emp = new Employee();
emp.PersonName = "Jones";
emp.Gender = "Male";
emp.EmpID = 201;
emp.Salary = 7000;
emp.Display();
System.Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/115. OOP - Interfaces 2/InterfaceExample2/InterfaceExample2/bin/Debug/InterfaceExempl...
Student Name: Allen
Gender: Male
Student ID: 101
Student Marks: 70

Employee Name: Jones
Gender: Male
Employee ID: 201
Employee Salary: 7000
```

Dynamic Polymorphism

Dynamic Polymorphism using interfaces

- Polymorphism means "decision making". The compiler takes a decision; which method has to be called. If the decision has been taken at compile-time (before running), we can call it as "compile-time polymorphism". If the decision would be taken at run-time (when we run the program), we can call it as "run-time polymorphism".
- "Method overloading" is an example of "compile-time polymorphism".
- We can implement dynamic polymorphism with interfaces or abstract classes.
- Steps implementation of dynamic polymorphism using interfaces:
 - First we have to create an interface with one or more methods.
 - Next we have implement it in two child classes.
 - Create a reference variable for the interface.
 - Create an object for first child class; Assign its reference into interface's variable & call the method now. It calls the first child class's method.
 - Create an object for second child class; Assign its reference into interface's variable & call the method now. It calls the second child class's method.
 - So, same statement is calling different methods in different situations, right! This is called "dynamic polymorphism".

Static Polymorphism (vs) Dynamic Polymorphism

Sl. No	Static Polymorphism	Dynamic Polymorphism
1	Also called as "compile-time polymorphism" or "early binding".	Also called as "run-time polymorphism" or "late binding".
2	Decision will be taken by the compiler at compile-time.	Decision will be taken by the CLR at run-time.

3	Implemented using "method overloading".	Implemented using "abstract classes" or "interfaces".
---	---	---

Dynamic Polymorphism - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DynamicPolymorphismExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DynamicPolymorphismExample”.
- Click on OK.

Program.cs

```
//interface
interface Interface1
{
    //abstract method in interface
    void Display();
}

//child class 1
class Class1 : Interface1
{
    //Overriding the "Interface1.Display" method
    public void Display()
```

```
{  
    System.Console.WriteLine("Class1.Display");  
}  
}  
  
//child class 2  
class Class2 : Interface1  
{  
    //Overriding the "Interface1.Display" method  
    public void Display()  
    {  
        System.Console.WriteLine("Class2.Display");  
    }  
}  
  
class Program  
{  
    static void Main()  
    {  
        //create a reference variable of interface type.  
        Interface1 i;  
        //create object for first child class  
        i = new Class1();  
        i.Display(); //It calls Class1.Display()  
        //create object for second child class  
        i = new Class2();  
        i.Display(); //It calls Class2.Display()  
        System.Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/116. OOP - Dynamic Polymorphism/DynamicPolymorphismExample/bin/Debug/Dynamic...
Class1.Display
Class2.Display
```

Dynamic Polymorphism – Example 2

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DynamicPolymorphismExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DynamicPolymorphismExample2”.
- Click on OK.

Program.cs

```
//interface
interface IPerson
{
    //properties
    string PersonName { get; set; }
    string Gender { get; set; }

    //abstract method
    void Display();
}

//child class 1
class Student : IPerson
```

```
{  
    //properties  
    public string PersonName { get; set; }  
    public string Gender { get; set; }  
  
    //implementing the IPerson.Display method  
    public void Display()  
    {  
        System.Console.WriteLine("Student Name: " + PersonName);  
        System.Console.WriteLine("Gender: " + Gender);  
    }  
}  
  
//child class 2  
class Employee : IPerson  
{  
    //properties  
    public string PersonName { get; set; }  
    public string Gender { get; set; }  
  
    //implementing the IPerson.Display method  
    public void Display()  
    {  
        System.Console.WriteLine("Employee Name: " + PersonName);  
        System.Console.WriteLine("Gender: " + Gender);  
    }  
}  
  
class Program  
{  
    static void Main()  
    {  
        //create reference variable of interface type  
        IPerson p;  
  
        //***** creating object for child class 1 *****/  
        p = new Student();
```

```
p.PersonName = "Allen";
p.Gender = "Male";
p.Display(); //calls Student.Display
System.Console.WriteLine("\n");

***** creating object for child class 2 *****
p = new Employee();
p.PersonName = "John";
p.Gender = "Male";
p.Display(); //calls Employee.Display
System.Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/117. OOP - Dynamic Polymorphism 2/DynamicPolymorphismExample2/DynamicPolymor...
Student Name: Allen
Gender: Male

Employee Name: John
Gender: Male
```

Multiple Inheritance using Interfaces

Multiple Inheritance using Interfaces

- Interface supports “multiple inheritance”. That means a child class can implement one or more interfaces. Then we must implement (define) all the methods of all the interfaces in the same child class.
- Then you can store the address of class’s object in any reference variable of any of those interfaces.

- Syntax for multiple inheritance:

```
class Classname: Interface1, Interface2, ...
{
    Implement all the methods or all interfaces here
}
```

Multiple Inheritance using Interfaces - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MultipleInheritanceExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “MultipleInheritanceExample”.
- Click on OK.

Program.cs

```
//interface 1
interface Interface1
{
    void Method1();
    void Method2();
}

//interface 2
interface Interface2
{
    void Method3();
    void Method4();
}

//child class - that implements multiple interfaces. This is called "multiple
inheritance".
class Class1 : Interface1, Interface2
{
    public void Method1()
    {
        System.Console.WriteLine("Class1.Method1");
    }

    public void Method2()
    {
        System.Console.WriteLine("Class1.Method2");
    }

    public void Method3()
    {
        System.Console.WriteLine("Class1.Method3");
    }
}
```

```
public void Method4()
{
    System.Console.WriteLine("Class1.Method4");
}
}

class Program
{
    static void Main()
    {
        //create reference variable & object for child class
        Class1 c1;
        c1 = new Class1();

        //create a reference variable for Interface1
        Interface1 i1;

        //store the address of child class's object in the interface reference
        //variable
        i1 = c1;

        //call the Interface1's methods only
        i1.Method1();
        i1.Method2();
        System.Console.WriteLine();

        //create a reference variable for Interface2
        Interface2 i2;

        //store the address of child class's object in the interface reference
        //variable
        i2 = c1;

        //call the Interface2's methods only
        i2.Method3();
        i2.Method4();
        System.Console.ReadKey();
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/118. OOP - Multiple Inheritance/MultipleInheritanceExample/bin/Debug/MultipleInheritan... — □ X
Class1.Method1
Class1.Method2

Class1.Method3
Class1.Method4
```

Interface Inheritance

Interface Inheritance

- An interface can inherit from another interface.
- Then the child class of child interface should implement all the methods of parent interface & child interface.

Syntax for interface inheritance:

```
interface ChildInterface: ParentInterface
{
}
```

Interface Inheritance - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InterfaceInheritanceExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InterfaceInheritanceExample”.
- Click on OK.

Program.cs

```
//parent interface
interface Interface1
```

```
{  
    void Method1();  
    void Method2();  
}  
  
//child interface  
interface Interface2 : Interface1  
{  
    void Method3();  
    void Method4();  
}  
  
//child class - that implements multiple interfaces. This is called "multiple  
inheritance".  
class Class1 : Interface2  
{  
    public void Method1()  
    {  
        System.Console.WriteLine("Class1.Method1");  
    }  
  
    public void Method2()  
    {  
        System.Console.WriteLine("Class1.Method2");  
    }  
  
    public void Method3()  
    {  
        System.Console.WriteLine("Class1.Method3");  
    }  
  
    public void Method4()  
    {  
        System.Console.WriteLine("Class1.Method4");  
    }  
}
```

```
class Program
{
    static void Main()
    {
        //create reference variable & object for child class
        Class1 c1;
        c1 = new Class1();

        //create a reference variable for Interface1
        Interface1 i1;

        //store the address of child class's object in the interface reference
        //variable
        i1 = c1;

        //call the Interface1's methods only
        i1.Method1();
        i1.Method2();
        System.Console.WriteLine();

        //create a reference variable for Interface2
        Interface2 i2;

        //store the address of child class's object in the interface reference
        //variable
        i2 = c1;

        //call the Interface2's methods only
        i2.Method3();
        i2.Method4();

        System.Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/119. OOP - Interface Inheritance/InterfaceInheritanceExample/bin/Debug/InterfaceInherita...
Class1.Method1
Class1.Method2

Class1.Method3
Class1.Method4
-
```

Sealed Classes

Sealed Classes

- Sealed classes can't be inheritable.
- If you don't want to allow your class to be inheritable by others, you can make it as sealed class.
- Sealed classes can contain any members just like normal classes.
- Syntax:

```
sealed class Classname
{
    Any members here
}
```

Normal Class (vs) Sealed Class

Sl. No	Normal Class	Sealed Class
1	Can be inherited.	Can't be inherited.

Sealed Classes - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SealedClassesExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “SealedClassesExample”.
- Click on OK.

Program.cs

```
//sealed class  
sealed class Class1  
{  
}  
  
//we can't inherit from "Class1"  
class Class2 : Class1  
{  
}  
  
class Program  
{  
    static void Main()  
    {  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

A screenshot of Microsoft Visual Studio 2022 showing a C# code editor. The code in `Program.cs` is as follows:

```
SealedClassesExample - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Toolbox Program.cs SealedClassesExample SealedClassesExample.Program Main()
namespace SealedClassesExample
{
    //sealed class
    sealed class Class1
    {
    }

    //we can't inherit from "Class1"
    class Class2 : Class1
    {
    }

    class Program
    {
        static void Main()
    }
}
```

The Error List window shows one error:

Code	Description	Project	File	Line
CS0509	'Class2': cannot derive from sealed type 'Class1'	SealedClassesExa	Program.cs	9

Note: You will get compile time error near “Class2”.

Namespaces

Namespaces

- Namespaces are collections of “types”, such as classes, structures, enumerations, interfaces and delegate types.
- The type that is present inside the namespace can be accessed by using “Namespacename.Typename” syntax.
- Namespaces are used to group-up the types that belongs to a module or a section of the project.
- Syntax:

```
namespace Namespacename  
{  
    Classes, Structures, interfaces, Enumerations here  
}
```

Namespaces - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “NamespacesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NamespacesExample”.
- Click on OK.

Program.cs

```
namespace Namespace1
{
    class Class1
    {
        public void Method1()
        {
            System.Console.WriteLine("Namespace1.Class1.Method1");
        }
    }
}
```

```
namespace NamespacesExample
{
    class Program
    {
        static void Main()
        {
            //create a reference variable
            Namespace1.Class1 c1;

            //create an object
            c1 = new Namespace1.Class1();

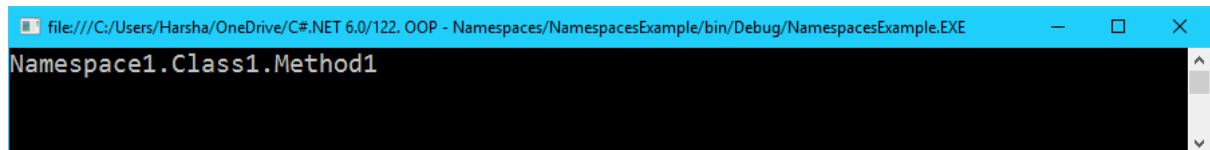
            //call the method
            c1.Method1();

            System.Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows command-line window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/122. OOP - Namespaces/NamespaceExample/bin/Debug/NamespaceExample.EXE". The window contains the text "Namespace1.Class1.Method1" and has standard window controls (minimize, maximize, close) at the top right.

HARSHA

Child Namespaces

Child Namespaces

- A namespace inside another namespace is called as “child namespace”.
- You can access the child namespace by using “ParentNamespace.ChildNamespace” syntax.
- You can access the types of child namespace by using “ParentNamespace.ChildNamespace.Typename” syntax.
- Syntax:

```
namespace ParentNamespaceName
{
    namespace ChildNamespaceName
    {
        Types (Classes, structures etc.)
    }
}
```

Child Namespaces - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ChildNamespacesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ChildNamespacesExample”.
- Click on OK.

Program.cs

```
//outer namespace starts
namespace ICICIBank
{
    //inner namespace starts
    namespace SavingsBanking
    {
        //class starts
        class BankAccount
        {
            public void Deposit()
            {

System.Console.WriteLine("ICICIBank.SavingsBanking.BankAccount.Deposit");
            }
        }
        //class ends
    }
    //inner namespace ends
}
//outer namespace ends

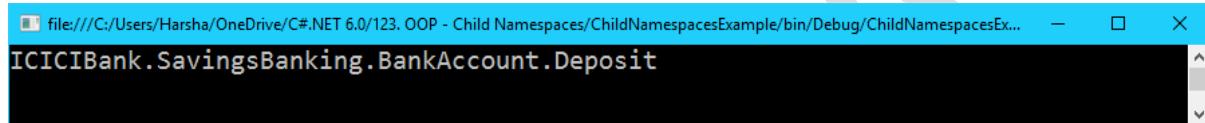
namespace ChildNamespacesExample
{
    class Program
    {
        static void Main()
        {
            //create a reference variable for BankAccount class
            ICICIBank.SavingsBanking.BankAccount c1;
            //create an object for BankAccount class
            c1 = new ICICIBank.SavingsBanking.BankAccount();
            //call the method
            c1.Deposit();
            System.Console.ReadKey();
        }
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Child Namespaces – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ChildNamespacesExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ChildNamespacesExample2”.
- Click on OK.

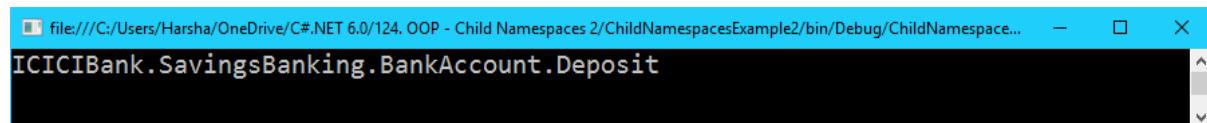
Program.cs

```
//Create a child namespace called "SavingsBanking" in a parent namespace  
//called "ICICIBank"  
namespace ICICIBank.SavingsBanking  
{  
    class BankAccount  
    {  
        public void Deposit()  
        {  
  
System.Console.WriteLine("ICICIBank.SavingsBanking.BankAccount.Depo  
sit");  
        }  
    }  
}  
  
namespace ChildNamespacesExample2  
{  
    class Program  
    {  
        static void Main()  
        {  
            //create reference variable  
ICICIBank.SavingsBanking.BankAccount c1;  
  
            //create object  
c1 = new ICICIBank.SavingsBanking.BankAccount();  
  
            //call method  
c1.Deposit();  
  
            System.Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“using” statement

Using

- The “using” statement is used to import a namespace in the file.
- When you import a namespace, you can access all its members (classes, structures etc.) directly, without writing the namespace name every time.
- Syntax:

using NamespaceName;

“using” statement - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “UsingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UsingExample”.
- Click on OK.

Program.cs

```
//import the namespace  
using ICICIBank.SavingsBanking;
```

```
//create a child namespace called "SavingsBanking" in a parent namespace  
//called "ICICIBank"  
namespace ICICIBank.SavingsBanking  
{  
    class BankAccount  
    {  
        public void Deposit()  
        {  
  
System.Console.WriteLine("ICICIBank.SavingsBanking.BankAccount.Depo  
sit");  
        }  
    }  
}
```

```
namespace UsingExample  
{  
    class Program  
    {  
        static void Main()  
        {  
            //create reference variable; access "BankAccount" class directly, as it  
is already imported above.  
            BankAccount c1;  
            //create object; access "BankAccount" class directly, as it is already  
imported above.  
            c1 = new BankAccount();  
            //call the method  
            c1.Deposit();  
            System.Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“using - alias” statement

Using Alias

- It is used to create a “nick name” (alias name) for a namespace, for easy access.
- You can access all the namespace’s members, by using the alias name.

Syntax:

```
using Aliasname = Namespacename;
```

“using - alias” statement - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “UsingAliasExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UsingAliasExample”.
- Click on OK.

Program.cs

```
//creating alias name
using i = India;
using p = Pakistan;

namespace India
{
    class Hyderabad
    {
        public void City()
        {
            System.Console.WriteLine("India.Hyderabad");
        }
    }
}

namespace Pakistan
{
    class Hyderabad
    {
        public void City()
        {
            System.Console.WriteLine("Pakistan.Hyderabad");
        }
    }
}

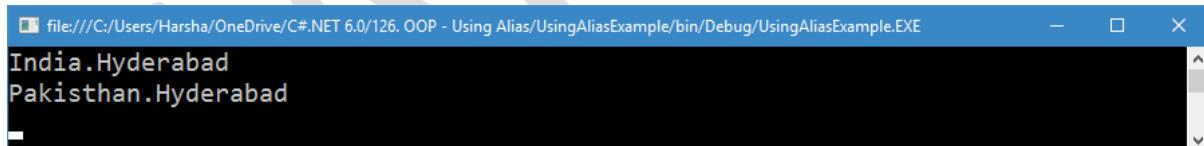
namespace UsingAliasExample
```

```
{  
    class Program  
    {  
        static void Main()  
        {  
            i.Hyderabad h1 = new i.Hyderabad();  
            p.Hyderabad h2 = new p.Hyderabad();  
            h1.City();  
            h2.City();  
            System.Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/126. OOP - Using Alias/UsingAliasExample/bin/Debug/UsingAliasExample.EXE". The window contains the following text:
India.Hyderabad
Pakistan.Hyderabad

Creating Separate Files for Classes

Separate Files

- In realtime applications, it is recommended to create a class in its own separate file, instead of creating all the classes in the same file.
- Class files make the large project easy to understand and maintain.
- Each class file's extension should be “.cs”.
- A project can have any no. of class files.
- To add a class file, open the solution explorer; right click on the project name and click on “Add” – “New Item”; select “Class”; type the class name as “Filename.cs”; click on “Add”.

Creating Separate Files for Classes - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SeparateFilesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SeparateFilesExample”. Click on OK.
- Open Solution Explorer. Right click on the project name (SeparateFilesExample) and click on “Add” – “New Item”. Click on “Code” – “Class”. Type the file name as “BankAccount.cs”. Click on “Add”. Then it creates the file “BankAccount.cs” in solution explorer.

BankAccount.cs

```
//Create a child namespace called "SavingsBanking" in a parent namespace  
//called "ICICIBank"  
namespace ICICIBank.SavingsBanking  
{  
    class BankAccount  
    {  
        public void Deposit()  
        {  
            System.Console.WriteLine("ICICIBank.SavingsBanking.  
BankAccount.Deposit");  
        }  
    }  
}
```

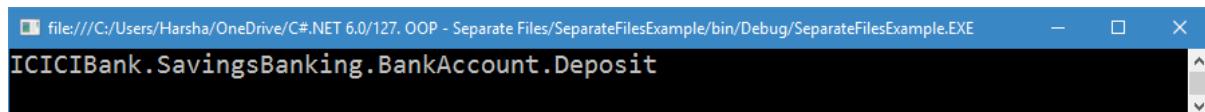
Program.cs

```
namespace mynamespace  
{  
    class Program  
    {  
        static void Main()  
        {  
            //create reference variable  
            ICICIBank.SavingsBanking.BankAccount c1;  
            //create object  
            c1 = new ICICIBank.SavingsBanking.BankAccount();  
            //call the method  
            c1.Deposit();  
            System.Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



HARSHA

Partial Classes

Partial Classes

- Partial classes are used to split a large class into multiple files.
- In real time, one part of the class will be created by one programmer and other part of the class will be created by other programmer; sometimes one part of the class will be automatically generated by visual studio and other part of the class will be created by the programmer.
- All the partial classes that should have a same name, should have “partial” keyword, and should be inside a same namespace.
- Syntax to create a partial class:

```
partial class Classname  
{  
    Members here  
}
```

Partial Classes - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “PartialClassesExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “PartialClassesExample”.
- Click on OK.
- Open Solution Explorer. Right click on the project and click on “Add” – “New Item”. Select “Class”. Type the file name as “File1.cs”. Click on “Add”.
- Open Solution Explorer. Right click on the project and click on “Add” – “New Item”. Select “Class”. Type the file name as “File2.cs”. Click on “Add”.

File1.cs

```
//create a parent namespace called "ICICIBank" and create a child
namespace called "SavingsBanking" in the parent namespace
namespace ICICIBank.SavingsBanking
{
    //create a class called "BankAccount" in "ICICIBank.SavingsBanking"
    namespace. This is a partial class. So at compilation time, the code for
    File2.cs also will be combined with this class.

    partial class BankAccount
    {
        //Fields
        public int accountnumber = 1234;
        public string accountholdername = "John";
        public double currentbalance = 10000;

        //Deposit method
        public void Deposit(double amount)
        {
            currentbalance = currentbalance + amount;
        }
    }
}
```

File2.cs

```
//create a parent namespace called "ICICIBank" and create a child
namespace called "SavingsBanking" in the parent namespace
namespace ICICIBank.SavingsBanking
{
    //Continue the definition of "BankAccount" class, which is defined at
    "File1.cs".
    partial class BankAccount
    {
        //Withdraw method
        public void Withdraw(double amount)
        {
            currentbalance = currentbalance - amount;
        }
    }
}
```

Program.cs

```
namespace PartialClassesExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable
            ICICIBank.SavingsBanking.BankAccount c1;

            //create object
            c1 = new ICICIBank.SavingsBanking.BankAccount();

            //get current balance
            System.Console.WriteLine(c1.currentbalance);
            System.Console.ReadKey();

            //deposit
        }
    }
}
```

```
c1.Deposit(5000);
System.Console.WriteLine(c1.currentbalance);
System.Console.ReadKey();

//withdraw
c1.Withdraw(2000);
System.Console.WriteLine(c1.currentbalance);

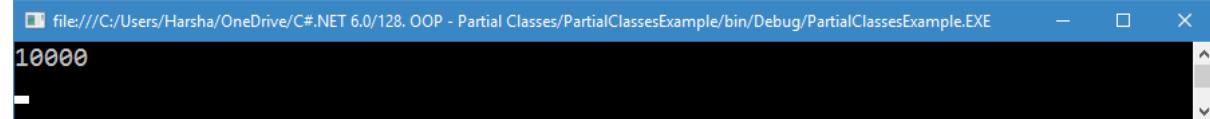
System.Console.ReadKey();
}

}
}
```

Running the Project

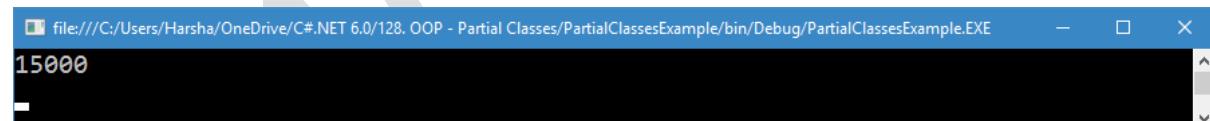
- Go to “Debug” menu and click on “Start Debugging”.

Output



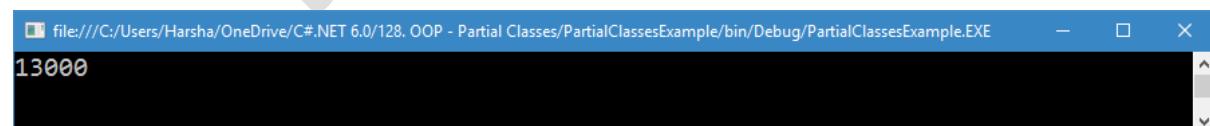
```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/128. OOP - Partial Classes/PartialClassesExample/bin/Debug/PartialClassesExample.EXE
10000
-
```

After Pressing Enter



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/128. OOP - Partial Classes/PartialClassesExample/bin/Debug/PartialClassesExample.EXE
15000
-
```

After Pressing Enter



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/128. OOP - Partial Classes/PartialClassesExample/bin/Debug/PartialClassesExample.EXE
13000
-
```

Enumerations

Enumerations

- Enumerations are used to avoid the entry of wrong values into the variable. It allows any one of the specific list of values.
- Enumeration is a collection of constants; each constant is assigned to an integer value automatically (Starts with zero (0) by default).
- Based on enumeration, we can create enumeration variables (as a Field or local variable).
- Enumeration variables can store any one of the enumeration constant.
- Syntax to create enumeration:

```
enum Enumerationname
{
    Constant1, Constant2, ...
}
```

Enumerations - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “EnumerationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EnumerationExample”.
- Click on OK.

Program.cs

```
using System;
//create a namespace called "EnumerationExample"
namespace EnumerationExample
{
    //create an enumeration called "GenderEnumeration" inside a namespace
    //called "EnumerationExample"
    enum GenderEnumeration
    {
        Male, Female
    }
    //create a class called "Person"
    class Person
    {
        //Field of "String" data type
        public string PersonName;

        //Field of "GenderEnumeration" data type. This Field can accept any
        one of the constants defined at GenderEnumeration only.
        public GenderEnumeration Gender;
    }

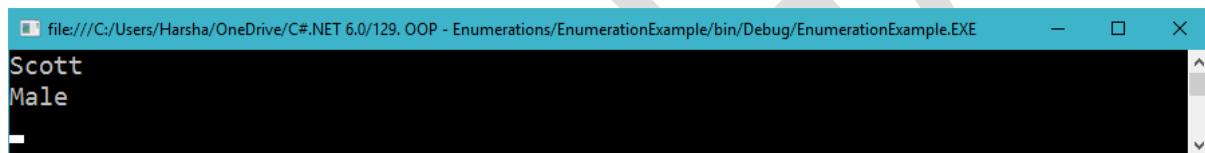
    class Program
    {
        static void Main()
        {
            //create reference variable of Person type
            Person p;
            //create an object of Person type
            p = new Person();
            //set data into PersonName
            p.PersonName = "Scott";
            //set data into "Gender". It accepts "Male" or "Female" only, as it is
            //GenderEnumeration" type.
            p.Gender = GenderEnumeration.Male;
            //get data from Fields
            Console.WriteLine(p.PersonName);
        }
    }
}
```

```
        Console.WriteLine(p.Gender);
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/129. OOP - Enumerations/EnumerationExample/bin/Debug/EnumerationExample.EXE". The window contains the following text:
Scott
Male
-

Structures

Structures

- Structure is type (data type), which is a collection of members such as Fields, methods, constructors, properties etc., similar to classes.
- No memory will be allocated for structure. We can create "structure variables" (instance) based on structure. For structure variables only memory will be allocated. "Structure variable" stores actual data.
- Syntax to create structure:

```
struct Structurename
{
    Fields, methods, constructors, properties etc.
}
```

Similarities between Classes and Structures

Sl. No	Class	Structure
1	Class is also a collection of members like Fields, methods, constructors and properties etc.	Structure is a collection of members like Fields, methods, constructors and properties etc.
2	A class is a data type.	A structure is a data type.

Differences between Classes and Structures

Sl. No	Class	Structure
1	An instance of class s called as "object". "Object" are stored in "heap".	"Instance of structure" is called as "structure variable". "Structure variables" are stored in "stack".
2	We can create a reference variable based on class.	We can't create a reference variable based on structure.

3	Classes are “reference-type” data types; because “objects” are stored in heap and its reference (address) will be stored in stack.	Structures are “value-type” data types; because “structure variables” (data) directly will be stored in stack; there is no “reference” (address) concept for structures.
4	Classes are little bit slower than structures; because to reach an object, we have to go through stack and find out the object based on the address stored in the reference variable.	Structures are little bit faster than classes; because to reach a structure variable, there is no need to search for the address.
5	Classes support “inheritance” and many other concepts.	Structures doesn’t support inheritance and its related concepts.
6	Classes are suitable for creating complex data structures & methods.	Structures are suitable for creating simple data structures.
7	“String”, “StringBuilder” etc., are pre-defined classes.	All standard data types (except “string”), such as int, float, double, DateTime etc., are structures.
8	Classes supports both “parameter-less constructor” and also “parameterized constructor”.	Structures doesn’t support “parameter-less constructor”; but supports “parameterized constructor”.
9	The following syntax creates a reference variable for the class: Classname referencevariable;	The following syntax creates a structure variable (instance) for the structure. Structurename Structurevariable;

10	<p>The following syntax creates an object (instance) for the class & also initializes all the Fields of the object:</p> <pre><code>new classname();</code></pre> <p>If the Field is numerical data type, it initializes zero (0). If the Field is string data type, it initializes null. If the Field is bool data type, it initializes 'false'.</p>	<p>The following syntax will not create an instance, but initializes all the Fields of structure variable (instance).</p> <pre><code>new structurename();</code></pre> <p>If the Field is numerical data type, it initializes zero (0). If the Field is string data type, it initializes null. If the Field is bool data type, it initializes 'false'.</p>
----	--	--

Structures - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StructuresExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StructuresExample”. Click on OK.

Program.cs

```
using System;
```

```
namespace StructuresExample
```

```
{
```

```
//create structure called "Student"
```

```
struct Student
{
    //Fields in structure
    public int studentid;
    public string studentname;
    public int marks;
}

class Program
{
    static void Main()
    {
        //create structure variable (instance)
        Student s;

        //set data into Fields of structure variable
        s.studentid = 1;
        s.studentname = "Scott";
        s.marks = 70;

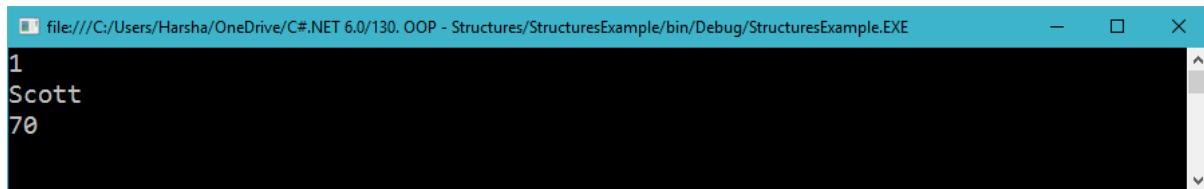
        //get data from Fields of structure variable
        Console.WriteLine(s.studentid);
        Console.WriteLine(s.studentname);
        Console.WriteLine(s.marks);

        Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/130. OOP - Structures/StructuresExample/bin/Debug/StructuresExample.EXE
1
Scott
70
```

Structures with Constructors - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StructuresWithConstructorsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StructuresWithConstructorsExample”.
- Click on OK.

Program.cs

```
using System;

namespace StructuresWithConstructorsExample
{
    //creating a structure
    struct Student
    {
        //Fields in structure
```

```

public int studentid;
public string studentname;
public int marks;
public string grade;

//constructor in structure
public Student(int value1, string value2, int value3)
{
    studentid = value1;
    studentname = value2;
    marks = value3;
    grade = "none";
}

//method in structure
public void CalculateGrade()
{
    if (marks >= 80 && marks <= 100)
        grade = "A grade";
    else if (marks >= 60 && marks < 80)
        grade = "B grade";
    else if (marks >= 50 && marks < 60)
        grade = "C grade";
    else if (marks >= 35 && marks < 50)
        grade = "D grade";
    else if (marks < 35)
        grade = "Fail";
}
}

class Program
{
    static void Main()
    {
        //create structure variable (instance)
        Student s;
    }
}

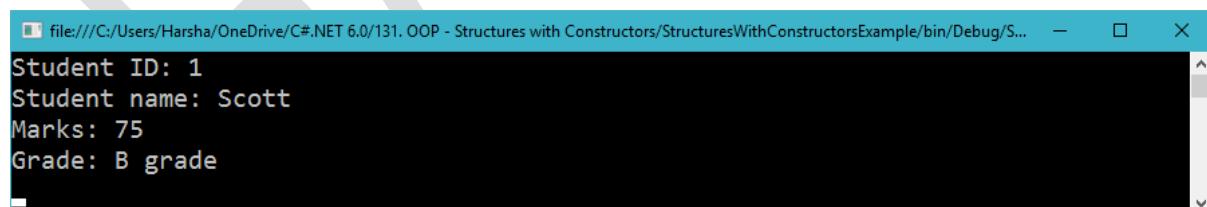
```

```
//call the constructor  
s = new Student(1, "Scott", 75);  
  
//get data from Fields of structure variable  
Console.WriteLine("Student ID: " + s.studentid);  
Console.WriteLine("Student name: " + s.studentname);  
Console.WriteLine("Marks: " + s.marks);  
  
//call method  
s.CalculateGrade();  
Console.WriteLine("Grade: " + s.grade);  
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/131. OOP - Structures with Constructors/StructuresWithConstructorsExample/bin/Debug/S... — □ ×  
Student ID: 1  
Student name: Scott  
Marks: 75  
Grade: B grade
```

Standard Data Types

Standard Data Types are Structures / Classes

- All the standard data types are structures / classes.

Sl. No	Standard Data Type	Structure / Class
1	sbyte	struct SByte
2	byte	struct Byte
3	short	struct Int16
4	ushort	struct UInt16
5	int	struct Int32
6	uint	struct UInt32
7	long	struct Int64
8	ulong	struct UInt64
9	float	struct Single
10	double	struct Double
11	decimal	struct Decimal
12	char	struct Char
13	string	class String
14	bool	struct Boolean
15	DateTime	struct DateTime

The "System.Object" class

The “object” data type & “System.Object” class

- The "object" data type is an alias (nick) name of "System.Object" class.
- The "System.Object" is ultimate base class in .net, based on which all structures and classes are inherited from.
- When you create a class in .net, automatically it will be treated as child class of the parent class called "System.Object".

"System.Object" Class - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ObjectClassExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ObjectClassExample”.
- Click on OK.

Program.cs

```
using System;
```

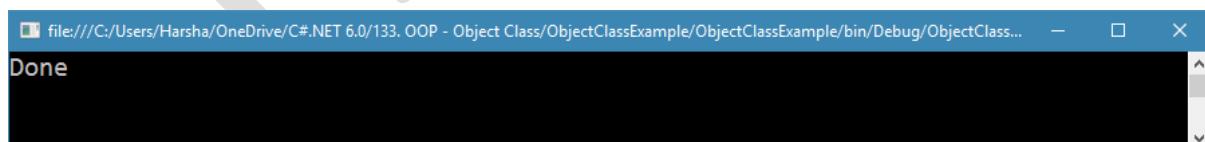
```
namespace ObjectClassExample
{
    class Class1
```

```
{  
}  
class Program  
{  
    static void Main()  
    {  
        //create a reference variable of "System.Object" type  
        System.Object a;  
  
        //create an object of "Class1" and store its address in the reference  
        //variable  
        a = new Class1();  
  
        Console.WriteLine("Done");  
  
        Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Methods of "System.Object" class

Methods of "System.Object" class

- The "System.Object" class is a parent class to all classes and structures in .net.
- The "System.Object" class provides the following methods to all classes and structures:

A. bool Equals(Other Object)

- The Equals() method compares the actual object and other object whether both are equal; It returns true if both are equal. It returns false if both are not equal.

B. int GetHashCode()

- The GetHashCode() method returns the memory address of the actual object.

C. Type GetType()

- The GetType() method returns the class name or structure name to which the actual object belongs to.

D. string ToString()

- The ToString() method returns the class name or structure name to which the actual object belongs to (similar to GetType method). But it can be overridden in the child class. If it is overridden, the child class's method will run.

Methods of “System.Object” class - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ObjectClassMethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ObjectClassMethodsExample”.
- Click on OK.

Program.cs

```
using System;

namespace ObjectClassMethodsExample
{
    // "Class1" is child class of "System.Object" class automatically
    class Class1
    {
        //override the ToString() method, which is the virtual method in
        System.Object class
        public override string ToString()
        {
            System.Console.WriteLine(base.GetHashCode());
            return "Class1.ToString";
        }
    }

    class Program
    {
```

```
static void Main()
{
    //create two reference variables for Class1
    Class1 c1, c2;

    //create two objects for Class1
    c1 = new Class1();
    c2 = new Class1();

    //call Equals method. This method compares whether c1 and c2 are
    equal.
    bool b = c1.Equals(c2);
    Console.WriteLine(b);

    //call GetHashCode method. This method returns the memory
    address of the object.
    int c = c1.GetHashCode();
    Console.WriteLine(c);

    //call GetType method. This method returns the type (class name) of
    the object.
    string t = c1.GetType().ToString();
    Console.WriteLine(t);

    //call ToString method. This method will call "ToString()" method,
    which is defined in "Class1".
    string s = c1.ToString();
    Console.WriteLine(s);

    Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/134. OOP - Object Class Methods/ObjectClassMethodsExample/bin/Debug/ObjectClassM... — X
False
37121646
ObjectClassMethodsExample.Class1
37121646
Class1.ToString
-
```

Boxing

Boxing

- It is a process of converting a value from "value-type data type (any structure)" to "reference-type data type (System.Object class)".
- It will be done automatically.

Syntax: *ReferenceTypeVariable = ValueTypeVariable;*

Boxing - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “BoxingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “BoxingExample”.
- Click on OK.

Program.cs

```
using System;

namespace BoxingExample
{
    class Program
    {
        static void Main()
```

```

{
    //creating a variable of "int" data type. "int" is a "structure". All the
    structures "value-type data types".
    int a = 1000;

    //creating a reference variable of "System.Object" data type.
    "System.Object" is a class. All the classes are "reference-type data types".
    System.Object obj;

    //Automatic conversion from "int" structure to "System.Object" class.
    This is called boxing.
    obj = a;

    //get values
    Console.WriteLine(a); //Output: 1000
    Console.WriteLine(obj); //Output: 1000

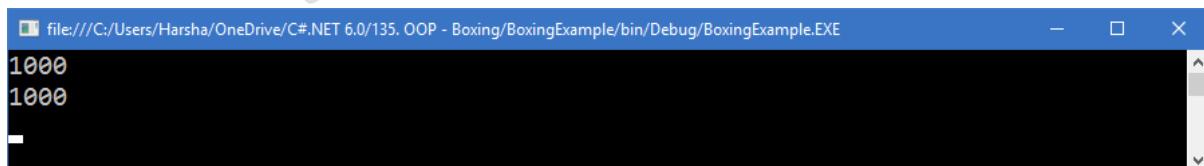
    Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/135. OOP - Boxing/BoxingExample/bin/Debug/BoxingExample.EXE
1000
1000
-
```

Unboxing

Unboxing

- It is a process of converting a value from "reference-type data type (System.Object)" to "value-type data type (any structure)". It should be done manually, by using "type casting" syntax.

Syntax: *ValueTypeVariable = (ExpectedDataTypehere)ReferenceTypeVariable;*

Unboxing - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Console Application".
- Type the project name as "UnboxingExample".
- Type the location as "C:\CSharp".
- Type the solution name as "UnboxingExample".
- Click on OK.

Program.cs

```
using System;

namespace UnboxingExample
{
    class Program
    {
```

```

static void Main()
{
    //create a reference variable of "System.Object" data type.
    "System.Object" is a class; All the classes are "reference-type" data type.
    Object obj = 100;

    //create a variable of "int" data type. "int" is a structure. All the
    structures "value-type" data type.
    int a;

    //manual conversion from "System.Object" class to "int" structure.
    This is called "unboxing".
    a = (int)obj;

    //get values
    Console.WriteLine(obj); //Output: 100
    Console.WriteLine(a); //Output: 100

    Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/136. OOP - Unboxing/UnboxingExample/bin/Debug/UnboxingExample.EXE
100
100
-

```

Static Constructors

Static Constructors

- Non-static constructors (normal constructors) initialize non-static Fields and static Fields; Static constructors initialize static Fields only. Static constructors will be called automatically when the class is used for the first time in Main method.
- Constructors are two types:
 - Non-static constructors (or) instance constructors
 - Static constructors (or) shared constructors

Sl. No	Non-Static Constructors	Static Constructors
1	By default, all the constructors are non-static constructors. These are not created with a keyword called "static".	Static constructors are created using "static" keyword.
2	Non-static constructors can access both non-static members and non-static members.	Static constructors can access only static members. Static constructors can't access non-static members. However, if you create an object for the class in the static constructor, then you can access any non-static members through the object, in the static constructor.
3	Non-static constructors are accessible with object only. When you create an object for the class, it calls the non-static constructor automatically. Non-static constructors are not accessible without creating an object for the class.	Static constructor will be called automatically when you access the class name for the first time in the Main method.

4	Non-static constructor will be called every time when we create an object for the class.	Static constructor will be called only once in the entire program.
5	We can use “this” keyword in the non-static constructors, because non-static constructors are called with an object.	We can’t use “this” keyword in the static constructor, because static constructor is called with class only, without an object.
6	A class can have ‘n’ no. of non-static constructors.	A class can have only one static constructor.
7	Use non-static constructors, if you want to initialize non-static Fields.	Use static constructor if you want to initialize static Fields.
8	Non-static constructors can receive ‘n’ no. of arguments.	Static constructor can’t receive any arguments.
9	Non-static constructors can’t return any value.	Static constructor also can’t return any value.
10	Access modifier such as “internal” and “public” can be applicable for the non-static constructor.	Access modifier is not applicable for the static constructor. By default, they are “public”.

Static Constructors - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “StaticConstructorsExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “StaticConstructorsExample”. Click on OK.

Program.cs

```
using System;

namespace StaticConstructorsExample
{
    //creating a class called "Class1"
    class Class1
    {
        //static Fields
        public static int a, b;

        //static constructor. It initializes the static Fields
        static Class1()
        {
            a = 10;
            b = 20;
        }
    }

    class Program
    {
        static void Main()
        {
            //Note: static constructor will be automatically called here

            //get data from static Fields
            Console.WriteLine(Class1.a); //Output: 10
            Console.WriteLine(Class1.b); //Output: 20

            Console.ReadKey();
        }
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/138. OOP - Static Constructors/StaticConstructorsExample/bin/Debug/StaticConstructorsE... 10 20
```

Static Classes

Static Classes

- Static classes can contain only static members (static Fields, static methods, static constructors, static properties etc.).
- Static classes can't contain non-static members.
- We can't create an object for static class.
- Syntax to create static class:

```
static class Classname
{
    Only static members here
}
```

Normal Class (vs) Static Class

Sl. No	Normal class	Static class
1	No "static" keyword.	Should have "static" keyword.
2	Can have both non-static members and static members also.	Can have only static members.
3	We can create an object for normal class.	We can't create an object for static class.

Static Classes - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StaticClassesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StaticClassesExample”. Click on OK.

Program.cs

using System;

```
namespace StaticClassesExample
{
    //static class. A static class contains only static Fields, static properties,
    static constructors, and static methods only
    static class Class1
    {
        //static Field
        public static string _Message;

        //static properties
        public static string Message
        {
            set
            {
                _Message = value;
            }
            get
            {
                return _Message;
            }
        }
    }

    //static constructor
    static Class1()
    {
        Message = "Hello";
    }
}
```

```

}

//static method
public static void Method1()
{
    Console.WriteLine("Class1.Method1");
}

class Program
{
    static void Main()
    {
        //Note: .NET automatically allocated memory for static Field here
        //Note: static constructor will be called automatically here

        //get value from static property
        Console.WriteLine(Class1.Message);

        //call static method
        Class1.Method1();

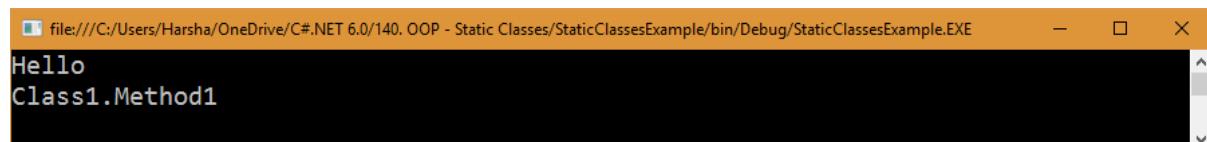
        Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Generic Classes

- Generic class is a class, which contains one or more generic-type Fields.
- Generic classes let you to store different values with respective to different objects in the same Fields.
- Generic classes let you to decide the data type of a Field, while creating object, rather than deciding the data type while creating the class.
- While creating an object for the generic class, you have to specify the data type of a Field.
- While creating a re-usable class, instead of you decide the data type of a Field within your class, if you want to let other programmers decide the data type that they want, you can make your class as generic class and make your Field as generic Field.

Syntax to create generic class:

```
class Classname<T>
{
    public T Fieldname;
}
```

Generic Classes - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “GenericsExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “GenericsExample”.
- Click on OK.

Program.cs

```
using System;

namespace GenericsExample
{
    //generic class with a generic-type parameter called "T"
    class Class1<T>
    {
        //generic Field of generic-type called "T"
        public T x;
    }

    class Program
    {
        static void Main()
        {
            //creating first generic object. As per this object "x" is of "int" data
            type.
            Class1<int> c1;
            c1 = new Class1<int>();
            c1.x = 100;
            Console.WriteLine(c1.x); //Output: 100

            //creating second generic object. As per this object "x" is of "string"
            data type.
            Class1<string> c2;
            c2 = new Class1<string>();
            c2.x = "hello";
            Console.WriteLine(c2.x); //Output: hello

            Console.ReadKey();
        }
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Multiple Generics - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MultipleGenericsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultipleGenericsExample”.
- Click on OK.

Program.cs

```

using System;

namespace MultipleGenericsExample
{
    //generic class with two generic-type parameters called "T1" and "T2"
    class Class1<T1, T2>
    {
        //generic Field of generic-type called "T1"
        public T1 x;

        //generic Field of generic-type called "T2"
        public T2 y;
    }

    class Program
    {
        static void Main()
        {
            //creating first generic object. As per this object "x" is of "int" data
            type and "y" is of "bool" data type.
            Class1<int, bool> c1;
            c1 = new Class1<int, bool>();
            c1.x = 100;
            c1.y = true;
            Console.WriteLine(c1.x); //Output: 100
            Console.WriteLine(c1.y); //Output: true
            Console.WriteLine();

            //creating second generic object. As per this object "x" is of "string"
            data type and "y" is of "double" data type.
            Class1<string, double> c2;
            c2 = new Class1<string, double>();
            c2.x = "hello";
            c2.y = 100.438;
            Console.WriteLine(c2.x); //Output: hello
            Console.WriteLine(c2.y); //Output: 100.438
        }
    }
}

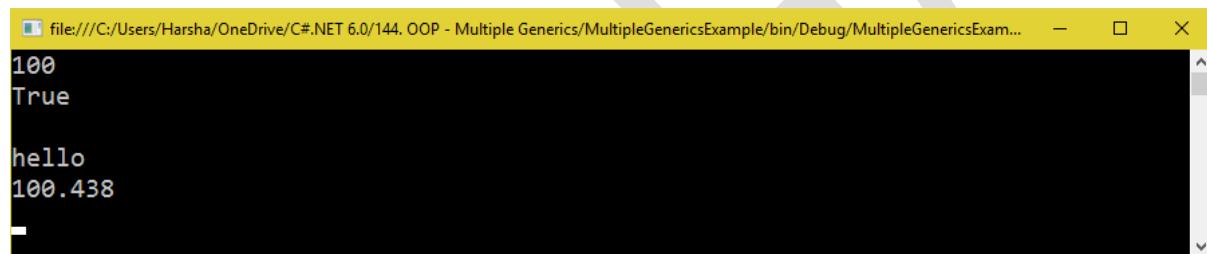
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/144. OOP - Multiple Generics/MultipleGenericsExample/bin/Debug/MultipleGenericsExam... — □ X
100
True

hello
100.438
-
```

Arrays

Arrays

- An array is a collection of multiple values of same data type.
- For each element, system automatically maintains an index. The index always starts from zero (0).
- Ex: Country names, City names, student marks etc.
- Syntax to create an array:

```
Datatype [ ] Arrayname = new Datatype [int length];
```

Arrays - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysExample”.
- Click on OK.

Program.cs

```
using System;
```

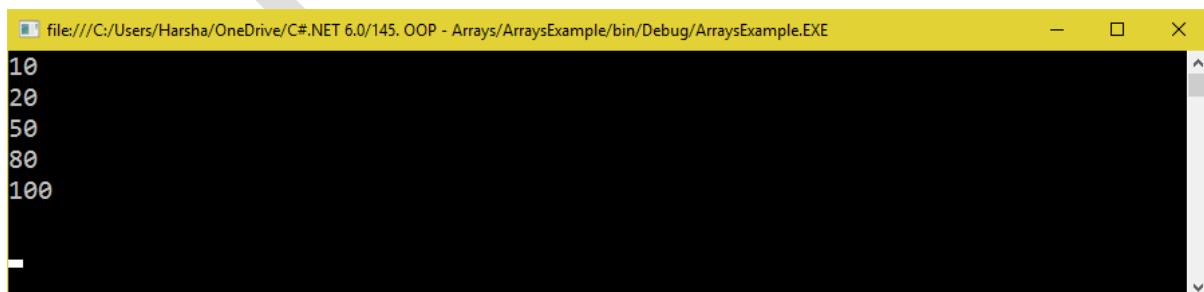
```
namespace ArraysExample
```

```
{  
    class Program  
    {  
        static void Main()  
        {  
            //create an array with 5 elements.  
            int[] a = new int[5] { 10, 20, 50, 80, 100 };  
  
            //get values from array, based on the index  
            Console.WriteLine(a[0]);  
            Console.WriteLine(a[1]);  
            Console.WriteLine(a[2]);  
            Console.WriteLine(a[3]);  
            Console.WriteLine(a[4]);  
            Console.ReadLine();  
  
            Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/145. OOP - Arrays/ArraysExample/bin/Debug/ArraysExample.EXE  
10  
20  
50  
80  
100  
-
```

Arrays with For Loop

Arrays with for loop

- We can read elements from an array, by using for loop:
- The for loop executes once for one element.

Syntax to read all elements from an array using for loop:

```
for (int i=0; i < arrayname.Length; i++)  
{  
    Code here  
}
```

Arrays with For Loop - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithForExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithForExample”.
- Click on OK.

Program.cs

```
using System;

namespace ArraysWithForExample
{
    class Program
    {
        static void Main()
        {
            //create an array with 5 elements
            int[] a = new int[5] { 10, 20, 50, 80, 100 };

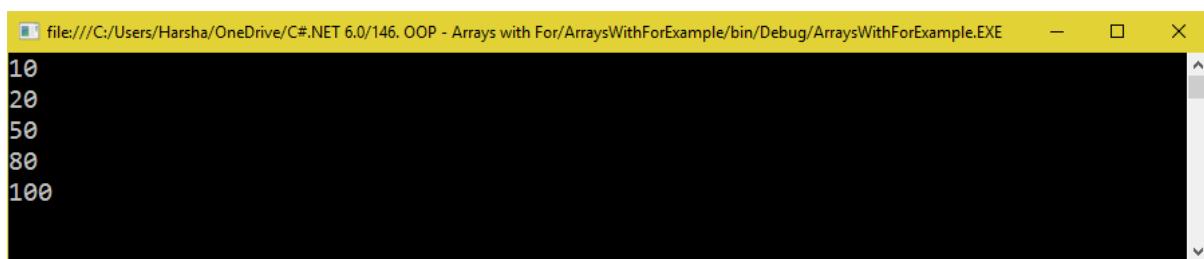
            //get values from array, using for loop
            for (int i = 0; i < a.Length; i++)
            {
                Console.WriteLine(a[i]);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/146. OOP - Arrays with For/ArraysWithForExample/bin/Debug/ArraysWithForExample.EXE
10
20
50
80
100
```

Arrays with Foreach Loop in C#.NET

Arrays with Foreach loop

- “foreach” loop is used to read all the values from an array.
- It executes once for one element in the array. Each time, the current element will be stored in the temporary variable.

Syntax of foreach loop:

```
foreach (Datatype Variablename in Arrayname)  
{  
}
```

- But it is used to read all the values of an array, sequentially. It is not possible to read some of the values, by using “foreach” loop. It is also not possible to read the elements randomly or in reverse order.

Arrays with Foreach Loop – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithForEachExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithForEachExample”.

- Click on OK.

Program.cs

```
using System;

namespace ArraysWithForEachExample
{
    class Program
    {
        static void Main()
        {
            //create an array with 5 elements
            int[] a = new int[5] { 10, 20, 50, 80, 100 };

            //get values from array, using foreach loop
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/147. OOP - Arrays with ForEach/ArraysWithForEachExample/bin/Debug/ArraysWithForEachExample.exe
10
20
50
80
100
-
```

“System.Array” class

“System.Array” class

- The “System.Array” is a pre-defined class. An object of “System.Array” class represents an array.
- In c#.net, every array is automatically treated as an object of “System.Array” class. So we can call methods of “System.Array” class, by using any array.
- The “System.Array” class has the following methods:
 1. System.Array.IndexOf()
 2. System.Array.BinarySearch()
 3. System.Array.Clear()
 4. System.Array.Resize()
 5. System.Array.Sort()
 6. System.Array.Reverse()
 7. System.Array.CopyTo()

“System.Array.IndexOf” method

Sl. No	Method
1	<p><code>int System.Array.IndexOf(System.Array array, object value)</code></p> <p>This method searches the array for the given value.</p> <p>If the value is found, it returns its index.</p> <p>If the value is not found, it returns -1.</p> <p>array: This parameter represents the array, in which you want to search.</p> <p>value: This parameter represents the actual value that is to be searched.</p> <p>The “IndexOf” method performs linear search. That means it searches all the elements of an array, until the search value is found. When the search value is found in the array, it stops searching and returns its index.</p> <p>The linear search has good performance, if the array is small. But if the array is larger, Binary search is recommended to improve the performance.</p>

“System.Array.IndexOf” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ArraysWithIndexOfExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithIndexOfExample”.
- Click on OK.

Program.cs

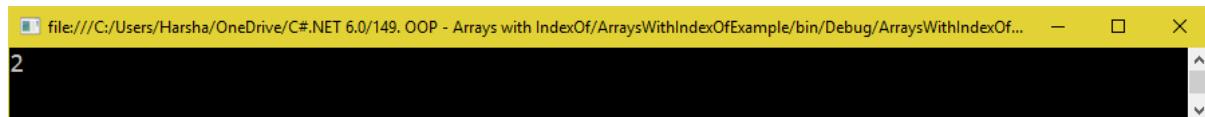
```
using System;
namespace ArraysWithIndexOfExample
{
    class Program
    {
        static void Main()
        {
            //create array with
            int[] a = new int[5] { 100, 2000, 50, 8, 1000 };

            //search (linear search) for "50" in array & get its index
            int n = Array.IndexOf(a, 50);
            Console.WriteLine(n); //Output: 2
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.Array.IndexOf" method – with NotFound – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithIndexOfWithNotFound”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithIndexOfNotFound”.
- Click on OK.

Program.cs

```
using System;
namespace ArraysWithIndexOfWithNotFound
{
    class Program
    {
        static void Main()
        {
            //create array
            int[] a = new int[5] { 100, 2000, 50, 8, 1000 };
```

//search (linear search) for "500" in array & get its index. If not found, it returns -1

```
int n = Array.IndexOf(a, 500);
Console.WriteLine(n);

Console.ReadKey();
}

}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/150. OOP - Arrays with IndexOf with NotFound/ArraysWithIndexOfWithNotFound/bin/Debug/arrayswithindexofwithnotfound.exe
-1
```

“System.Array.BinarySearch” method

Sl. No	Method
2	<p><code>int System.Array.BinarySearch(System.Array array, object value)</code></p> <p>This method searches the array for the given value.</p> <p>If the value is found, it returns its index.</p> <p>If the value is not found, it returns -1.</p> <p>array: This parameter represents the array, in which you want to search.</p> <p>value: This parameter represents the actual value that is to be searched.</p> <p>The “BinarySearch” method performs binary search. The “Binary Search” requires an array, which is already sorted. On unsorted arrays, binary search is not possible.</p> <p>Binary search means it directly goes to the middle of the array (array size / 2), and checks that item is less than / greater than the search value. If that item is greater than the search value, it searches only in the first half of the array. If that item is less than the search value, it searches only in the second half of the array. Thus it searches only half of the array. So in this way, it saves performance.</p>

“System.Array.BinarySearch” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ArraysWithBinarySearchExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithBinarySearchExample”.
- Click on OK.

Program.cs

```
using System;

namespace ArraysWithBinarySearchExample
{
    class Program
    {
        static void Main()
        {
            //For Binary search, we must need a sorted array (should have the
            values in ascending order)
            int[] a = new int[5] { 10, 20, 50, 80, 100 };

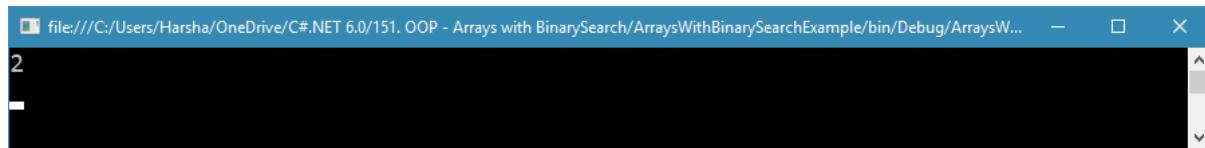
            //search (binary search) for "50" in sorted array & get its index
            int n = Array.BinarySearch(a, 50);
            Console.WriteLine(n);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
2
```

“System.Array.Clear” method

Sl. No	Method
3	<p><code>void System.Array.Clear(System.Array array, int index, int length)</code></p> <p>This method starts with the given index and sets all the “length” no. of elements to zero (0).</p> <p>array: This parameter represents the array, in which you want to clear the elements.</p> <p>index: This parameter represents the index, from which clearing process is to be started.</p> <p>length: This parameter represents the no. of elements that are to be cleared.</p>

“System.Array.Clear” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithClearExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithClearExample”.
- Click on OK.

Program.cs

```
using System;

namespace ArraysWithClearExample
{
    class Program
    {
        static void Main()
        {
            //create array
            int[] a = new int[8] { 10, 20, 50, 80, 100, 780, 900, 1000 };

            //set the 3 elements to zero (0), starting from index "2"
            Array.Clear(a, 2, 3);

            //get values from array
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/152. OOP - Arrays with Clear/ArraysWithClearExample/bin/Debug/ArraysWithClearExempl... 10 20 0 0 0 780 900 1000
```

“System.Array.Resize” method

Sl. No	Method
4	<p><code>void System.Array.Resize(ref System.Array array, int newSize)</code></p> <p>This method increases / decreases size of the array.</p> <p>array: This parameter represents the array, which you want to resize.</p> <p>newSize: This parameter represents the new size of the array, how many elements you want to store in the array. It can be less than or greater than the current size.</p>

“System.Array.Resize” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithResizeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithResizeExample”.
- Click on OK.

Program.cs

```
using System;
```

```
namespace ArraysWithResizeExample
{
    class Program
    {
        static void Main()
        {
            //create array (current size of array is "4")
            int[] a = new int[4] { 10, 20, 50, 80 };

            //change size of the array from "4" to "6"
            Array.Resize(ref a, 6);
            a[4] = 200;
            a[5] = 350;

            //get values from array
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/153. OOP - Arrays with Resize/ArraysWithResizeExample/bin/Debug/ArraysWithResizExam... - X
```

```
10  
20  
50  
80  
200  
350  
-
```

“System.Array.Sort” method

Sl. No	Method
5	<p>void System.Array.Sort(System.Array array)</p> <p>This method sorts the array in ascending order.</p> <p>array: This parameter represents the array, which you want to sort.</p>

“System.Array.Sort” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithSortExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithSortExample”.
- Click on OK.

Program.cs

```
using System;
```

```
namespace ArraysWithSortExample
```

```
{
```

```
    class Program
```

```
{  
    static void Main()  
    {  
        //create array  
        int[] a = new int[7] { 50, 20, 100, 30, 500, 150, 300 };  
  
        //sort the array  
        Array.Sort(a);  
  
        //get values from array  
        foreach (int item in a)  
        {  
            Console.WriteLine(item);  
        }  
  
        Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window with the title bar "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/154. OOP - Arrays with Sort/ArraysWithSortExample/bin/Debug/ArraysWithSortExample.EXE". The window displays the following text:
20
30
50
100
150
300
500
-

“System.Array.Reverse” method

Sl. No	Method
6	<p>void System.Array.Reverse(System.Array array)</p> <p>This method reverses the array.</p> <p>array: This parameter represents the array, which you want to reverse.</p>

“System.Array.Reverse” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithReverseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithReverseExample”.
- Click on OK.

Program.cs

```
using System;

namespace ArraysWithReverseExample
{
```

```
class Program
{
    static void Main()
    {
        //create array
        int[] a = new int[7] { 50, 20, 100, 30, 500, 150, 300 };

        //reverse the array
        Array.Reverse(a);

        //get values from the array
        foreach (int item in a)
        {
            Console.WriteLine(item);
        }
        Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/155. OOP - Arrays with Reverse/ArraysWithReverseExample/bin/Debug/ArraysWithReverse...
300
150
500
30
100
20
50
```

“System.Array.CopyTo” method

Sl. No	Method
7	<p><code>void sourceArray.CopyTo(System.Array destinationArray, int startIndex)</code></p> <p>This method copies all the elements from source array to destination array.</p> <p>sourceArray: This parameter represents the array, which array you want to copy.</p> <p>destinationArray: This parameter represents the array, into which you want to copy the elements of source array.</p> <p>startIndex: This parameter represents the index of the element, from which you want to start copying.</p>

“System.Array.CopyTo” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ArraysWithCopyToExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArraysWithCopyToExample”.

- Click on OK.

Program.cs

```
using System;

namespace ArraysWithCopyToExample
{
    class Program
    {
        static void Main()
        {
            //create array
            int[] a = new int[5] { 10, 20, 50, 80, 100 };

            //create another array
            int[] b = new int[5];

            //copy all the elements from one array to another array
            a.CopyTo(b, 0);

            //get values from array
            foreach (int item in b)
            {
                Console.WriteLine(item);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/156. OOP - Arrays with CopyTo/ArraysWithCopyToExample/bin/Debug/ArraysWithCopyTo...
10
20
50
80
100
-
```

Multidimensional Array

Multi-dimensional array

- Multi-dimensional arrays are used to store multiple values in a single row of an array.
- Syntax to create multi-dimensional array:

Datatype[,] Arrayname = new Datatype[int rowLength, int columnLength]

- Note: You can increase the no. of commas (,) as many as you want.

Multidimensional Array - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “MultiDimArraysExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultiDimArraysExample”.
- Click on OK.

Program.cs

```
using System;

namespace MultiDimArraysExample
{
    class Program
    {
        static void Main()
```

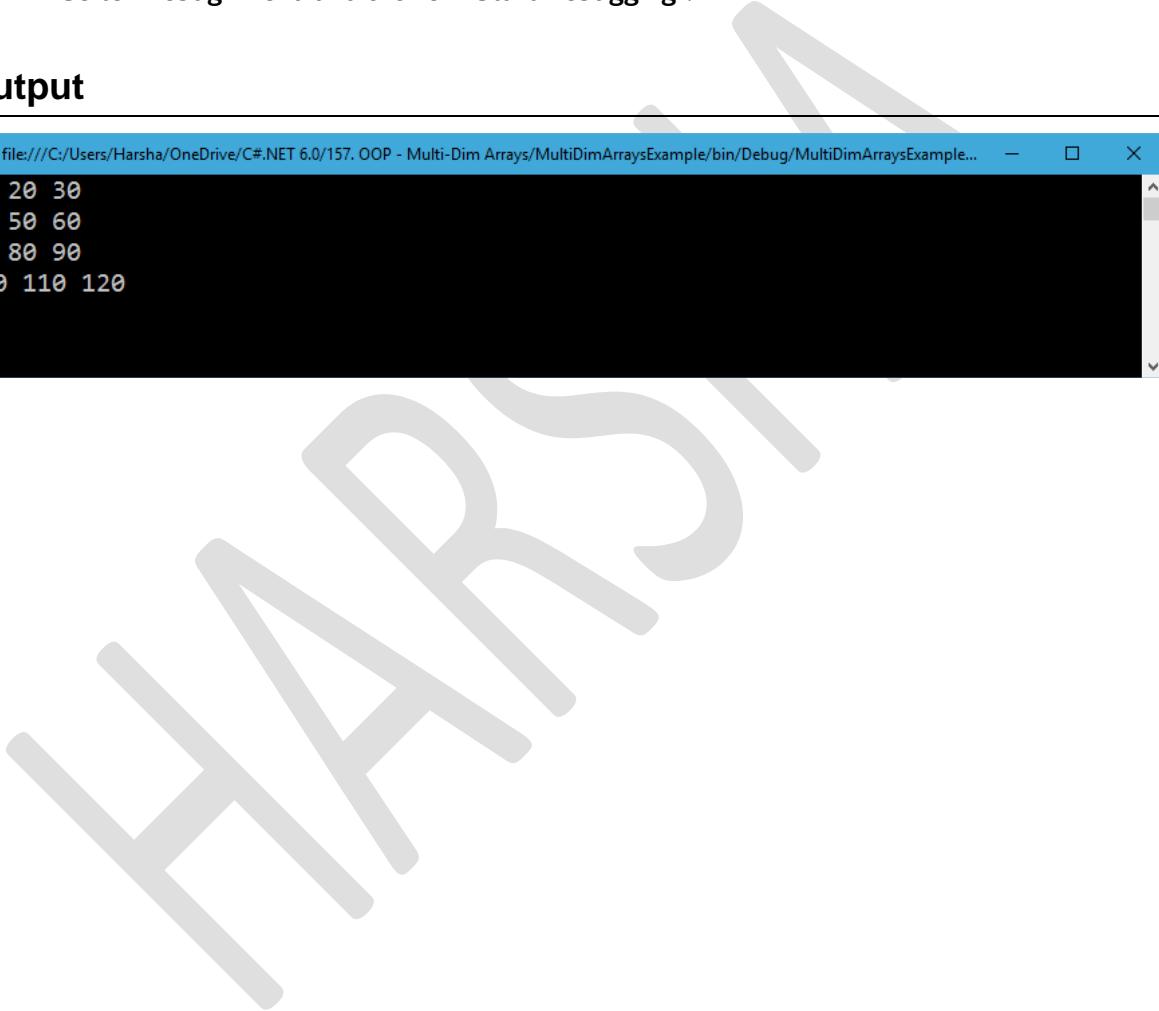
```
{  
    //create multi-dim array  
    int[,] a = new int[4, 3]  
    {  
        { 10, 20, 30 },  
        { 40, 50, 60 },  
        { 70, 80, 90 },  
        { 100, 110, 120 }  
    };  
  
    //get first row from multi-dim array  
    Console.Write(a[0, 0]);  
    Console.Write(" ");  
    Console.Write(a[0, 1]);  
    Console.Write(" ");  
    Console.WriteLine(a[0, 2]);  
  
    //get second row from multi-dim array  
    Console.Write(a[1, 0]);  
    Console.Write(" ");  
    Console.Write(a[1, 1]);  
    Console.Write(" ");  
    Console.WriteLine(a[1, 2]);  
  
    //get third row from multi-dim array  
    Console.Write(a[2, 0]);  
    Console.Write(" ");  
    Console.Write(a[2, 1]);  
    Console.Write(" ");  
    Console.WriteLine(a[2, 2]);  
  
    //get fourth row from multi-dim array  
    Console.Write(a[3, 0]);  
    Console.Write(" ");  
    Console.Write(a[3, 1]);  
    Console.Write(" ");  
    Console.WriteLine(a[3, 2]);
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/157. OOP - Multi-Dim Arrays/MultiDimArraysExample/bin/Debug/MultiDimArraysExample...
10 20 30
40 50 60
70 80 90
100 110 120
-
```

Collections

Limitations of arrays

- Adding or removing elements from array (without a duplicate copy) is not possible.
- Solution: Use collections.

What are collections

- A collection is a "group of values" of same data type.
- We can add, remove values in the collection anywhere within the program.
- Collections internally use arrays, managed by some pre-defined methods.
- To create collections .net provides a set of collection classes.

List of collections classes in .NET:

1. System.Collections.Generic.List
2. System.Collections.Generic.Dictionary
3. System.Collections.Generic.SortedList
4. System.Collections.Hashtable
5. System.Collections.ArrayList

Arrays (vs) Collections

Sl. No	Arrays	Collections
1	Array is treated as an object of "System.Array" class.	Collection is treated as an object of collection-based class such as "System.Collections.Generic.List",

		"System.Collections.Generic.Dictionary" etc.
2	Doesn't support adding, removing elements dynamically.	Support adding, removing elements dynamically.
3	Arrays is old concept.	Collections are recommended and mostly-used in real time.
4	If you want to store fixed no. of values, then use "arrays".	If you want to add or remove elements later, then use "collections".

The "List" class

System.Collections.Generic.List

- "List" is a class, in "System.Collections.Generic" namespace.
- This class's object represents a collection.
- "List" is a collection-based class.
- List is used to store multiple values of same data type.
- We can add or remove elements in the collection dynamically.
- It is a generic class; it has a generic type argument called "T". That means you must pass a data type as argument to it, while creating an object.

The "List" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".

- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ListExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListExample”.
- Click on OK.

Program.cs

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

namespace ListExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 90, 100 };

            //get values from list collection
            Console.WriteLine(a[0]);
            Console.WriteLine(a[1]);
            Console.WriteLine(a[2]);
            Console.WriteLine(a[3]);
            Console.WriteLine(a[4]);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/159. OOP - List/ListExample/bin/Debug/ListExample.EXE
10
40
50
90
100
```

The "List" class – with “for” loop

List with for loop

- We can read elements from a list collection, by using for loop:
- The “for loop” executes once for one element.
- Syntax to read all elements from the list collection, using for loop:

```
for(int i = 0; i < collectionname.Count; i++)  
{  
    Code here  
}
```

The "List" class – with “for” loop - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ListWithForExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithForExample”.
- Click on OK.

Program.cs

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

```
namespace ListWithForExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 90, 100 };

            //get values from list collection using for loop
            for (int i = 0; i < a.Count; i++)
            {
                Console.WriteLine(a[i]);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/160. OOP - List with For/ListWithForExample/bin/Debug>ListWithForExample.EXE". The window displays the following text:
10
40
50
90
100
-
The window has standard minimize, maximize, and close buttons at the top right.

The "List" class – with “foreach” loop

List with Foreach loop

- “foreach” loop is used to read all the values from a list collection.
- It executes once for one element in the collection. Each time, the current element will be stored in the temporary variable.

Syntax of foreach loop:

```
foreach (Datatype Variablename in Collectionname)
{
}
```

- But it is used to read all the values of a collection, sequentially. It is not possible to read some of the values, by using “foreach” loop. It is also not possible to read the elements randomly or in reverse order.

The "List" class – with “foreach” loop - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ListWithForEachExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithForEachExample”.
- Click on OK.

Program.cs

```

using System;
using System.Collections.Generic;

namespace ListWithForEachExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 90, 100 };

            //get values from list collection using foreach loop
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }

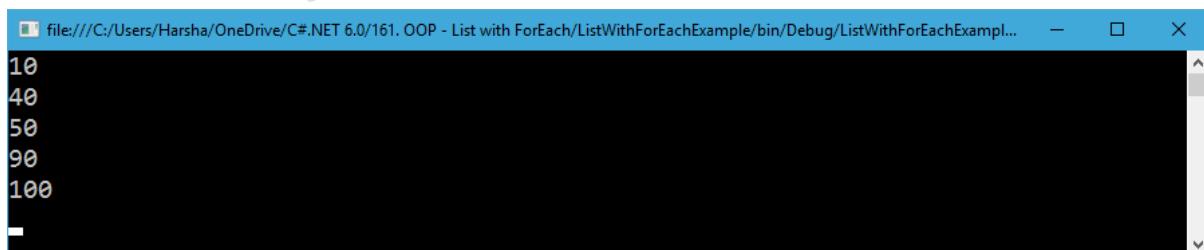
            Console.ReadKey();
        }
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

10
40
50
90
100

```

"System.Collections.Generic. List.Add" method

Sl. No	Method
1	<p>void Add(object value)</p> <p>This method adds a new value at the end of existing collection.</p> <p>value: This parameter represents the actual value that is to be added.</p>

"System.Collections.Generic. List.Add" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithAddExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithAddExample”.
- Click on OK.

Program.cs

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

namespace ListWithAddExample
```

```

{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50 };
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();
            //add "100" at the end of list collection
            a.Add(100);
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/162. OOP - List with Add/ListWithAddExample/bin/Debug/ListWithAddExample.EXE
10
40
50

10
40
50
100

```

"System.Collections.Generic.List.Insert" method

Sl. No	Method
2	<p>void Insert(int index, object value)</p> <p>This method inserts a new value at the specified index in the existing collection.</p> <p>index: This parameter represents the index, where the value is to be inserted.</p> <p>value: This parameter represents the actual value that is to be added.</p>

"System.Collections.Generic.List.Insert" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithInsertExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithInsertExample”.
- Click on OK.

Program.cs

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

```
namespace ListWithInsertExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 150, 200 };

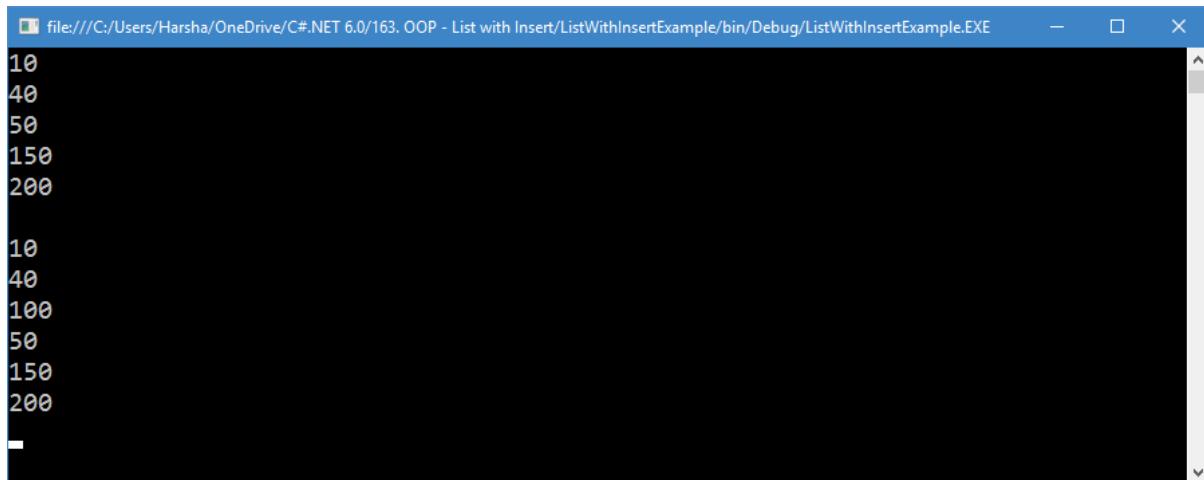
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();

            //insert the value "100" at the position number "2"
            a.Insert(2, 100);
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/163. OOP - List with Insert/ListWithInsertExample/bin/Debug/ListWithInsertExample.EXE
10
40
50
150
200

10
40
100
50
150
200
-
```

"System.Collections.Generic. List.AddRange" method

Sl. No	Method
3	<p><code>void AddRange(collection)</code></p> <p>This method adds an array or collection at the end of existing collection.</p> <p>collection: This parameter represents the array / collection that is to be added.</p>

"System.Collections.Generic. List.AddRange" method - Example

Creating Project

- Open Visual Studio 2019.

- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ListWithAddRangeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithAddRangeExample”.
- Click on OK.

Program.cs

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

namespace ListWithAddRangeExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 150, 200 };

            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();

            //create another list collection called "extra"
            List<int> extra = new List<int>() { 300, 400, 500 };

            //add all the elements of "extra" collection at the end of "a" collection
        }
    }
}
```

```
a.AddRange(extra);

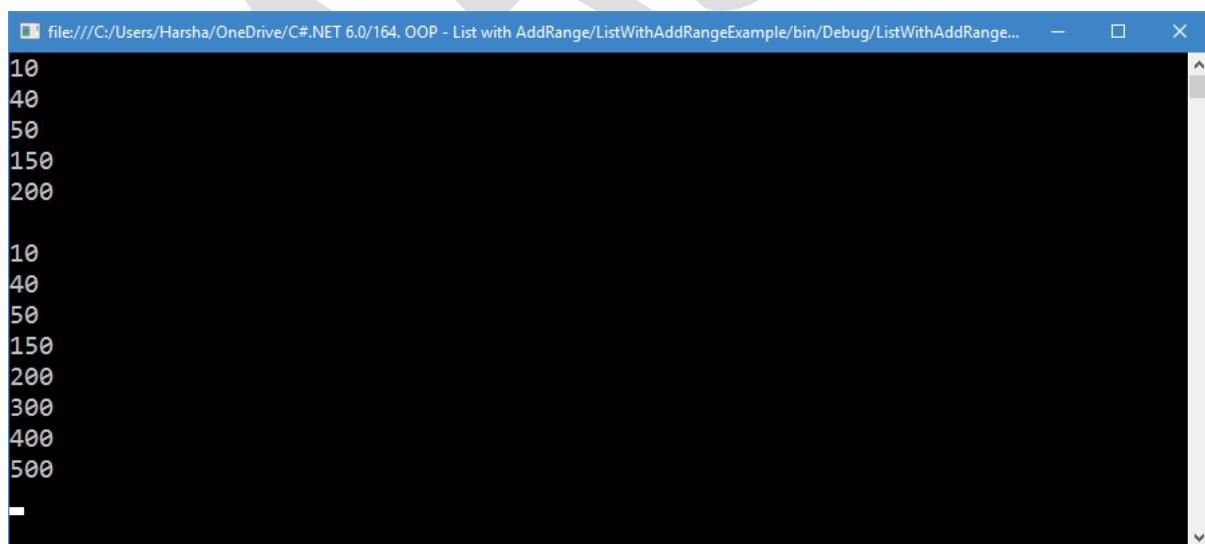
//get values from list collection
foreach (int item in a)
{
    Console.WriteLine(item);
}

Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/164. OOP - List with AddRange/ListWithAddRangeExample/bin/Debug>ListWithAddRange...
10
40
50
150
200

10
40
50
150
200
300
400
500
-
```

"System.Collections.Generic. List.InsertRange" method

Sl. No	Method
4	<p><code>void InsertRange(int index, collection)</code></p> <p>This method inserts an array or collection at the specified index in the existing collection.</p> <p>index: This parameter represents the index, where the array / collection is to be inserted.</p> <p>collection: This parameter represents the array / collection that is to be inserted.</p>

"System.Collections.Generic. List.InsertRange" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ListWithInsertRangeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithInsertRangeExample”.
- Click on OK.

Program.cs

```
using System;
using System.Collections.Generic;
namespace ListWithInsertRangeExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 150, 200 };
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();
            //create another collection called "extra"
            List<int> extra = new List<int>() { 60, 70, 80 };
            //insert all the elements of "extra" collection into "a" collection at
            position no. "3"
            a.InsertRange(3, extra);
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/165. OOP - List with InsertRange/ListWithInsertRangeExample/bin/Debug/ListWithInsertRa...
10
40
50
150
200

10
40
50
60
70
80
150
200
-
```

HARSH

"System.Collections.Generic.List.Remove" method in C#.NET

Sl. No	Method
5	<p><code>void Remove(object value)</code></p> <p>This method removes the specified value in the end of existing collection.</p> <p>value: This parameter represents the value that is to be removed.</p>

"System.Collections.Generic.List.Remove" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithRemoveExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithRemoveExample”.
- Click on OK.

Program.cs

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

```
namespace ListWithRemoveExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50 };

            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();

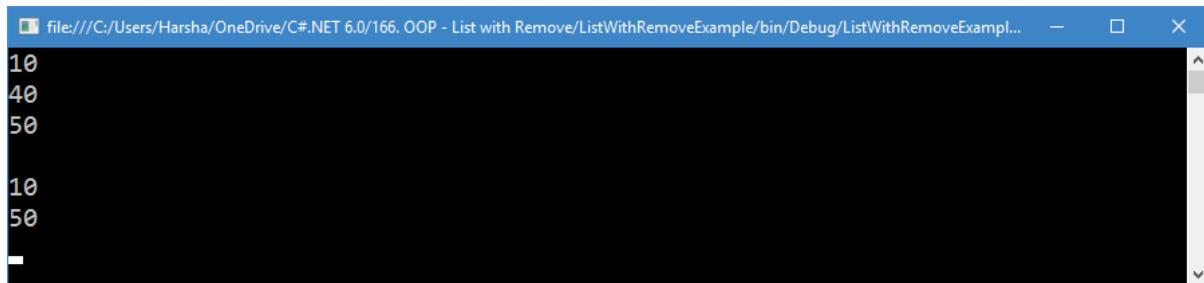
            //remove "40" from the list collection
            a.Remove(40);

            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/166. OOP - List with Remove/ListWithRemoveExample/bin/Debug/ListWithRemoveExampl...
10
40
50
10
50
-
```

"System.Collections.Generic. List.RemoveAt" method

Sl. No	Method
6	<p><code>void RemoveAt(int index)</code></p> <p>This method removes the existing value at the specified index in the existing collection.</p> <p>index: This parameter represents the index, where the value is to be removed.</p>

"System.Collections.Generic. List.RemoveAt" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithRemoveAtExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithRemoveAtExample”.
- Click on OK.

Program.cs

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

namespace ListWithRemoveAtExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 120, 200 };

            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();

            //remove the element at position no. "2" in the collection
            a.RemoveAt(2);

            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/167. OOP - List with RemoveAt/ListWithRemoveAtExample/bin/Debug/ListWithRemoveAt...
10
40
50
120
200

10
40
120
200
```

"System.Collections.Generic. List.Clear" method

Sl. No	Method
7	void Clear() This method deletes all the elements in the existing collection.

"System.Collections.Generic. List.Clear" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithClearExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithClearExample”.
- Click on OK.

Program.cs

```

using System;
using System.Collections.Generic;

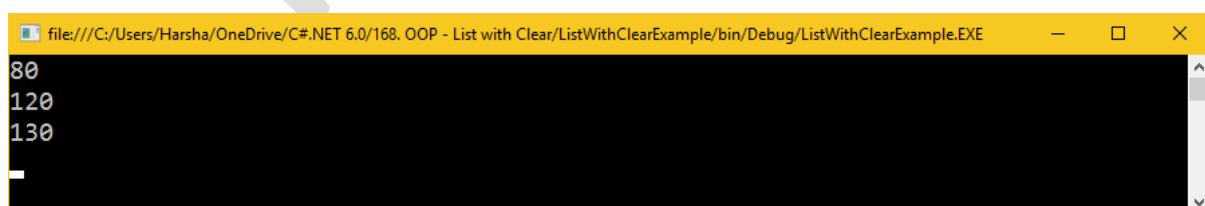
namespace ListWithClearExample
{
    class Program
    {
        static void Main()
    }
}
  
```

```
{  
    //create list collection  
    List<int> a = new List<int> { 10, 20, 50 };  
  
    //clear all items in the list collection  
    a.Clear();  
  
    //add new items into the list collection  
    a.Add(80);  
    a.Add(120);  
    a.Add(130);  
  
    //get values from list collection  
    foreach (int item in a)  
    {  
        Console.WriteLine(item);  
    }  
  
    Console.ReadKey();  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.Collections.Generic. List.IndexOf" method

Sl. No	Method
8	<p><code>int IndexOf(object value)</code></p> <p>This method searches the collection for the given value.</p> <p>If the value is found, it returns its index.</p> <p>If the value is not found, it returns -1.</p> <p>value: This parameter represents the actual value that is to be searched.</p> <p>The “IndexOf” method performs linear search. That means it searches all the elements of the collection, until the search value is found. When the search value is found in the collection, it stops searching and returns its index.</p> <p>The linear search has good performance, if the array is small. But if the collection is larger, “Binary search” is recommended to improve the performance.</p>

"System.Collections.Generic. List.IndexOf" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ListWithIndexOfExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithIndexOfExample”.
- Click on OK.

Program.cs

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

namespace ListWithIndexOfExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 120, 200 };

            //search for "50" in the list collection & get its index
            int n = a.IndexOf(50);
            Console.WriteLine(n);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/169. OOP - List with IndexOf/ListWithIndexOfExample/bin/Debug/ListWithIndexOfExample...
2
```

"System.Collections.Generic.List.IndexOf" method – Not Found- Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithIndexOfNotFoundExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithIndexOfNotFoundExample”.
- Click on OK.

Program.cs

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

namespace ListWithIndexOfNotFound
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>() { 10, 40, 50, 120, 200 };

            //search for "500" in the list collection & get its index
            int n = a.IndexOf(500);
            Console.WriteLine(n);

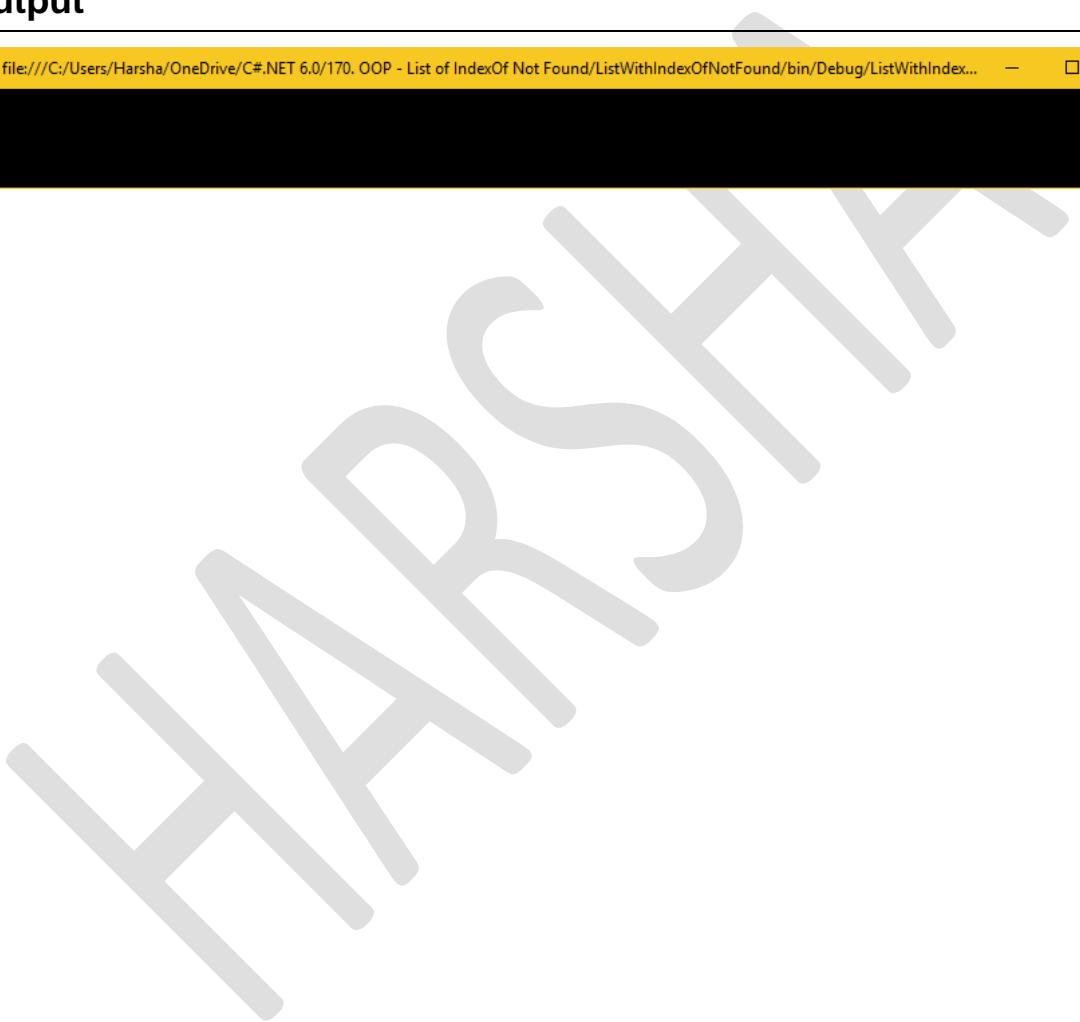
            Console.ReadKey();
        }
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/170. OOP - List of IndexOf Not Found/ListWithIndexOfNotFound/bin/Debug/ListWithIndex... - X
-1
```

"System.Collections.Generic. List.BinarySearch" method

Sl. No	Method
9	<p><code>int BinarySearch(object value)</code></p> <p>This method searches the collection for the given value.</p> <p>If the value is found, it returns its index.</p> <p>If the value is not found, it returns -1.</p> <p>value: This parameter represents the actual value that is to be searched.</p> <p>The “BinarySearch” method performs binary search. The “Binary Search” requires a collection, which is already sorted. On unsorted collection, binary search is not possible. Binary search means it directly goes to the middle of the collection (collection size / 2), and checks that item is less than / greater than the search value. If that item is greater than the search value, it searches only in the first half of the collection. If that item is less than the search value, it searches only in the second half of the collection. Thus it searches only half of the collection. So in this way, it saves performance.</p>

"System.Collections.Generic.List.BinarySearch" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#". Select "Console Application".
- Type the project name as "ListWithBinarySearchExample".
- Type the location as "C:\CSharp".
- Type the solution name as "ListWithBinarySearchExample".
- Click on OK.

Program.cs

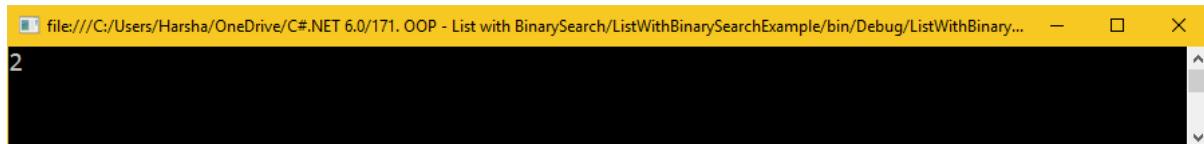
```
using System;
using System.Collections.Generic;
namespace ListWithBinarySearchExample
{
    class Program
    {
        static void Main()
        {
            //list collection with sorted items (values should in a ascending order)
            List<int> a = new List<int> { 10, 20, 50, 80, 100 };

            //search for "50" in the collection
            int n = a.BinarySearch(50);
            Console.WriteLine(n);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.Collections.Generic. List.Contains" method

Sl. No	Method
10	<p><code>void Contains(object value)</code></p> <p>This method searches the collection for the given value.</p> <p>If the value is found, it returns true.</p> <p>If the value is not found, it returns false.</p> <p>value: This parameter represents the actual value that is to be searched.</p>

"System.Collections.Generic. List.Contains" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithContainsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithContainsExample”.
- Click on OK.

Program.cs

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

namespace ListWithContainsExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>{ 10, 20, 50, 80, 100 };

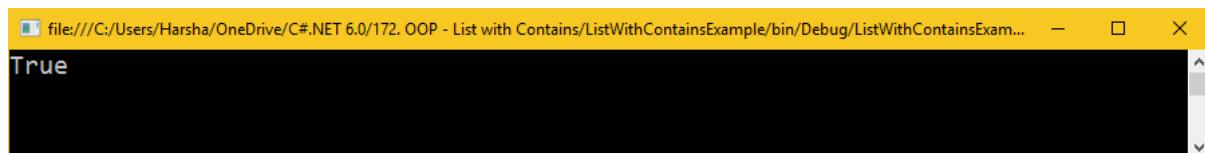
            //check whether the list has "50" or not
            bool b = a.Contains(50);
            Console.WriteLine(b);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/172. OOP - List with Contains/ListWithContainsExample/bin/Debug/ListWithContainsExam... - X
True
```

"System.Collections.Generic. List.Reverse" method

Sl. No	Method
12	void Reverse() This method reverses the collection.

"System.Collections.Generic. List.Reverse" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithReverseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithReverseExample”.

- Click on OK.

Program.cs

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

namespace ListWithReverseExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int> { 50, 20, 100, 30, 500, 150, 300 };

            //reverse the collection
            a.Reverse();

            //get values from the list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/173. OOP - List with Reverse/ListWithReverseExample/bin/Debug/ListWithReverseExample.... - X
300
150
500
30
100
20
50
-
```

"System.Collections.Generic. List.Sort" method

Sl. No	Method
11	<code>void Sort()</code> This method sorts the collection in ascending order.

"System.Collections.Generic. List.Sort" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithSortExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithSortExample”.

- Click on OK.

Program.cs

```
using System;
using System.Collections.Generic;
namespace ListWithSortExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>{ 50, 20, 100, 30, 500, 150, 300 };

            //sort collection (in ascending order)
            a.Sort();
            //get values from list collection
            foreach (int item in a)
            {
                Console.WriteLine(item);
            }
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
20
30
50
100
150
300
500
```

System.Collections.Generic. List - Sort Descending - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ListWithSortDescendingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListWithSortDescendingExample”.
- Click on OK.

Program.cs

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

namespace ListWithSortDescendingExample
{
```

```
class Program
{
    static void Main()
    {
        //create list collection
        List<int> a = new List<int>{ 50, 20, 100, 30, 500, 150, 300 };

        //sort collection (in descending order)
        a.Sort();
        a.Reverse();

        //get values from list collection
        foreach (int item in a)
        {
            Console.WriteLine(item);
        }

        Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/175. OOP - List with Sort Descending/ListWithSortDescendingExample/bin/Debug/ListWith... - □ X
500
300
150
100
50
30
20
```

Collection Filter

- To get elements based on the condition, write a foreach loop to iterate through each element, check the condition and add the matching element into another collection.

Collection Filter - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “CollectionFilterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CollectionFilterExample”.
- Click on OK.

Program.cs

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

namespace CollectionFilterExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>{ 50, 27, 105, 34, 55, 80, 100, 133 };

            //create an empty new collection
            List<int> b = new List<int>();
```

```
//read all the elements from "a" collection, check the condition, and  
add the matching elements into a new collection called "b"  
foreach (int item in a)  
{  
    if (item % 2 == 0)  
    {  
        b.Add(item);  
    }  
}  
  
//get values from the new collection  
foreach (int item in b)  
{  
    Console.WriteLine(item);  
}  
Console.ReadKey();  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/176. OOP - Collection Filter/CollectionFilterExample/bin/Debug/CollectionFilterExample.EXE  
50  
34  
80  
100
```

LINQ

Understanding LINQ

- LINQ stands for "Language Integrated Query".
- It is a concept to retrieve conditional data from an array or collection & store the matching elements in another collection.
- Syntax: `from variable in collection where condition select variable;`
 - The “from clause” represents the variable. Each value from the collection will be loaded into the variable; and then the where condition will be checked. If the condition is true, then the value will be added to the result collection.
 - The “in” clause represents the array / collection from which the data has to be fetched.
 - The “where” clause represents the condition to filter the data.
 - The “select” clause represents the value that has to be added to the result collection / array.

LINQ - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “LinqExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LinqExample”.
- Click on OK.

Program.cs

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

namespace LinqExample
{
    class Program
    {
        static void Main()
        {
            //create list collection
            List<int> a = new List<int>{ 50, 27, 105, 34, 55, 80, 100, 133 };

            //create an empty new collection
            List<int> b = new List<int>();

            //read all the elements from "a" collection, check the condition, and
            //add the matching elements into a new collection called "b" using LINQ
            // (Language Integrated Query)
            b = (from item in a where (item % 2 == 0) select item).ToList();

            //get values from the new collection
            foreach (int item in b)
            {
                Console.WriteLine(item);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/177. OOP - Linq/LinqExample/bin/Debug/LinqExample.EXE
50
34
80
100
```

Collection of Objects

- An object stores a set of Fields.
- A "collection of objects" is a set of objects. Ex: Employees, Students etc.

Collection of Objects - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “CollectionOfObjectsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CollectionOfObjectsExample”.
- Click on OK.

Program.cs

```
using System;
using System.Collections.Generic;
namespace CollectionOfObjectsExample
{
    class Student
    {
        public int StudentID { get; set; }
        public string StudentName { get; set; }
        public int Marks { get; set; }
    }
    class Program
    {
```

```

static void Main()
{
    //create 4 reference variables for Student class
    Student s1, s2, s3, s4;
    //create 4 objects for Student class
    s1 = new Student() { StudentID = 1, StudentName = "scott", Marks = 40 };
    s2 = new Student() { StudentID = 2, StudentName = "allen", Marks = 70 };
    s3 = new Student() { StudentID = 3, StudentName = "jones", Marks = 25 };
    s4 = new Student() { StudentID = 4, StudentName = "john", Marks = 50 };

    //create a collection of "Student" type
    List<Student> Students = new List<Student>();
    //add objects to collection
    Students.Add(s1);
    Students.Add(s2);
    Students.Add(s3);
    Students.Add(s4);

    //get objects from the collection
    for (int i = 0; i < Students.Count; i++)
    {
        Console.WriteLine(Students[i].StudentID);
        Console.WriteLine(",");
        Console.WriteLine(Students[i].StudentName);
        Console.WriteLine(",");
        Console.WriteLine(Students[i].Marks);
    }
    Console.ReadKey();
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/178. OOP - Collection of Objects/CollectionOfObjectsExample/bin/Debug/CollectionOfOb...
1, scott, 40
2, allen, 70
3, jones, 25
4, john, 50
```

Collection of Objects – Filter - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CollectionOfObjectsFilterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CollectionOfObjectsFilterExample”. Click on OK.

Program.cs

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

namespace CollectionOfObjectsFilterExample
{
    class Student
    {
        public int StudentID { get; set; }
        public string StudentName { get; set; }
        public int Marks { get; set; }
```

```
}
```

```
class Program
{
    static void Main()
    {
        //create 4 reference variables for Student class
Student s1, s2, s3, s4;

        //create 4 objects for Student class
s1 = new Student() { StudentID = 1, StudentName = "scott", Marks = 40 };
s2 = new Student() { StudentID = 2, StudentName = "allen", Marks = 25 };
s3 = new Student() { StudentID = 3, StudentName = "jones", Marks = 70 };
s4 = new Student() { StudentID = 4, StudentName = "john", Marks = 32 };

        //create collection of Student type and initialize objects into the
collection with collection initializer
List<Student> Students = new List<Student>() { s1, s2, s3, s4 };
        //create a new empty collection
List<Student> PassedStudents = new List<Student>();
        //read all the objects from "Students" collection, check the condition,
and add matching objects into "PassedStudents" collection
        for (int i = 0; i < Students.Count; i++)
        {
            Student s;
            s = Students[i];

            if (s.Marks >= 35)
                PassedStudents.Add(s);
        }
        //get objects from the "PassedStudents" collection
        for (int i = 0; i < PassedStudents.Count; i++)
        {
            Student s;
            s = PassedStudents[i];
            Console.WriteLine(s.StudentID);
            Console.Write(", ");
        }
    }
}
```

```
        Console.WriteLine(s.StudentName);
        Console.Write(",");
        Console.WriteLine(s.Marks);
    }
    Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/179. OOP - Collection of Objects Filter/CollectionOfObjectsFilterExample/bin/Debug/Colle...
1, scott, 40
3, jones, 70
```

Collection of Objects – LINQ - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CollectionOfObjectsLinqExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “CollectionOfObjectsLinqExample”.
- Click on OK.

Program.cs

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

namespace CollectionOfObjectsLinqExample
{
    class Student
    {
        public int StudentID { get; set; }
        public string StudentName { get; set; }
        public int Marks { get; set; }
    }
    class Program
    {
        static void Main()
        {
            //create a collection of "Student" type and initialize 4 objects
            List<Student> Students = new List<Student>()
            {
                new Student() { StudentID = 1, StudentName = "scott", Marks = 40 },
                new Student() { StudentID = 2, StudentName = "allen", Marks = 25 },
                new Student() { StudentID = 3, StudentName = "jones", Marks = 70 },
                new Student() { StudentID = 4, StudentName = "john", Marks = 32 }
            };

            //create a new empty collection
            List<Student> PassedStudents = new List<Student>();

            //read all the objects from "Students" collection, check the condition,
            and add matching objects into a new collection using LINQ
```

```

//PassedStudents = (from s in Students where s.Marks >= 35 select
s).ToList();
PassedStudents = Students.Where(s => s.Marks >= 35).ToList();

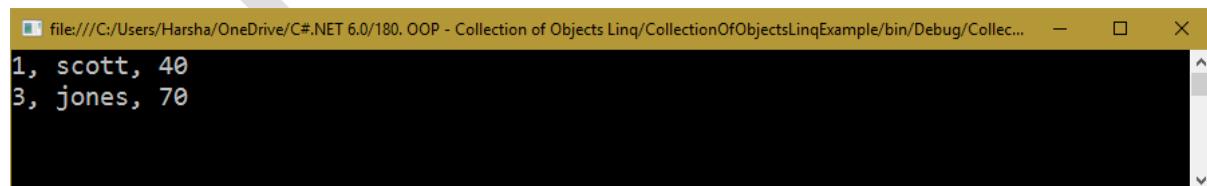
//get objects from "PasssedStudents" collection
for (int i = 0; i < PassedStudents.Count; i++)
{
    Student s;
    s = PassedStudents[i];
    Console.WriteLine(s.StudentID);
    Console.Write(",");
    Console.WriteLine(s.StudentName);
    Console.Write(",");
    Console.WriteLine(s.Marks);
}
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/180. OOP - Collection of Objects Linq/CollectionOfObjectsLinqExample/bin/Debug/Collect...
1, scott, 40
3, jones, 70

```

The "Dictionary" class

System.Collections.Generic.Dictionary

- "Dictionary" is a pre-defined class in "System.Collections.Generic" namespace.
- Dictionary is used to store a set of pairs of key and values.
- Example:

Key	Value
Maths	60
Physics	70
Chemistry	80

- It is also generic class that receives two data types; one is for key; other one is for value.
- We can get the value based on the key, rather than based on index.

The "Dictionary" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DictionaryExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DictionaryExample”.
- Click on OK.

Program.cs

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

namespace DictionaryExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable for dictionary collection
            Dictionary<string, int> Marks;

            //create object for dictionary collection
            Marks = new Dictionary<string, int>();

            //add items (keys and values)
            Marks.Add("Maths", 86);
            Marks.Add("Physics", 67);
            Marks.Add("Chemistry", 78);

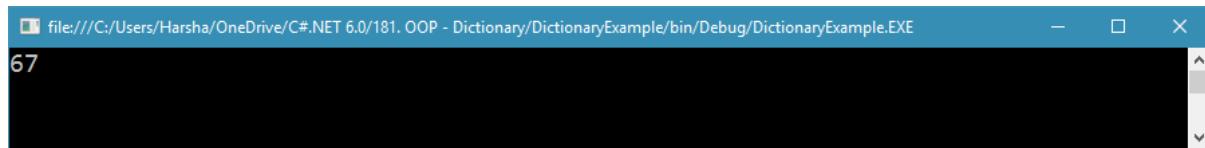
            //get value based on key
            int m = Marks["Physics"];
            Console.WriteLine(m);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "SortedList" class

System.Collections.Generic.SortedList

- "SortedList" is a pre-defined class in "System.Collections.Generic" namespace.
- SortedList is also used to store a set of pairs of key and values.
- It is also generic class that receives two data types; one is for key, other one is for value.
- It automatically sorts the data while adding elements itself. So that it will be faster while retrieving the values.
- We can get the value based on the key, rather than based on index.

The "SortedList" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “SortedListExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SortedListExample”.
- Click on OK.

Program.cs

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

namespace SortedListExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable for SortedList collection
            SortedList<string, int> Marks;

            //create object for SortedList collection
            Marks = new SortedList<string, int>();

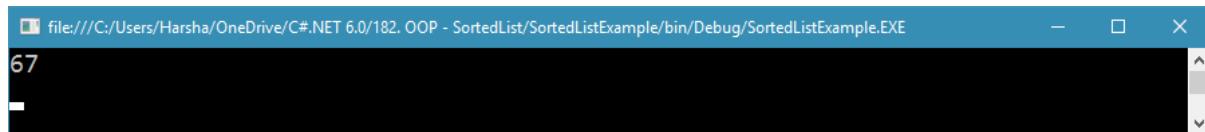
            //add items (keys and values)
            Marks.Add("Maths", 86);
            Marks.Add("Physics", 67);
            Marks.Add("Chemistry", 78);

            //get value based on the key
            int m = Marks["Physics"];
            Console.WriteLine(m);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
67
```

The "Hashtable" class in .NET

System.Collections.Hashtable

- "Hashtable" is a pre-defined class in "System.Collections" namespace.
- Hashtable is similar to Dictionary, but no need to specify the data types for key and value; so that the key and value can be of any data type.
- It is not a generic class.
- We can get the value based on the key, rather than based on index.

The "Hashtable" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#". Select "Console Application".
- Type the project name as "HashtableExample".
- Type the location as "C:\CSharp".
- Type the solution name as "HashtableExample".
- Click on OK.

Program.cs

```
using System;
using System.Collections;

namespace HashtableExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable for Hashtable collection
            Hashtable Marks;

            //create object for Hashtable collection
            Marks = new Hashtable();

            //add items (keys and values)
            Marks.Add("Maths", 86);
            Marks.Add("Physics", 67);
            Marks.Add("Chemistry", 78);
            Marks.Add("StudentName", "scott");
            Marks.Add("IsRegistered", true);

            //get value based on the key
            int m = Convert.ToInt32(Marks["Physics"]);
            Console.WriteLine(m);

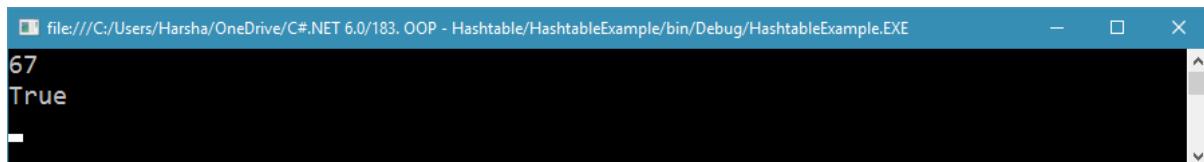
            //get value based on the key (another example)
            bool b = Convert.ToBoolean(Marks["IsRegistered"]);
            Console.WriteLine(b);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/183. OOP - Hashtable/HashtableExample/bin/Debug/HashtableExample.EXE
67
True
```

The "ArrayList" class in .NET

System.Collections.ArrayList

- “ArrayList” is a pre-defined class in “System.Collections” namespace.
- It is similar to List, but no need to specify the data type for value; so that the value can be of any data type.
- It is also not a generic class.

The "ArrayList" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “ArrayListExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ArrayListExample”.
- Click on OK.

Program.cs

```
using System;
using System.Collections;

namespace ArrayListExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable for ArrayList collection
            ArrayList MyData;
            //create object for ArrayList collection
            MyData = new ArrayList();
            //add items
            MyData.Add(100);
            MyData.Add(100.2398);
            MyData.Add("hyderabad");
            MyData.Add(true);
            MyData.Add(DateTime.Now);

            //get items from the ArrayList collection
            for (int i = 0; i < MyData.Count; i++)
            {
                Console.WriteLine(MyData[i]);
            }
            Console.ReadKey();
        }
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/184. OOP - ArrayList/ArrayListExample/bin/Debug/ArrayListExample.EXE
100
100.2398
hyderabad
True
9/16/2016 4:45:10 PM
```

The "typeof" operator in .NET

"typeof" operator:

- The "typeof" operator is used to check the data type of a value.
- Syntax: `typeof (data type name)`

The "typeof" operator - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “TypeofExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TypeofExample”.
- Click on OK.

Program.cs

```
using System;
using System.Collections;

namespace TypeofExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable for ArrayList collection
            ArrayList MyData;

            //create object for ArrayList collection
            MyData = new ArrayList();

            //add items
            MyData.Add(100);
            MyData.Add(150);
            MyData.Add(100.2398);
            MyData.Add("hyderabad");
            MyData.Add(true);
            MyData.Add(DateTime.Now);

            //get items from the ArrayList collection
            for (int i = 0; i < MyData.Count; i++)
        }
    }
}
```

```
{  
    if (MyData[i].GetType() == typeof(int))  
    {  
        int n = Convert.ToInt32(MyData[i]);  
        Console.WriteLine(n);  
    }  
    else if (MyData[i].GetType() == typeof(double))  
    {  
        double n = Convert.ToDouble(MyData[i]);  
        Console.WriteLine(n);  
    }  
    else if (MyData[i].GetType() == typeof(string))  
    {  
        string n = Convert.ToString(MyData[i]);  
        Console.WriteLine(n);  
    }  
    else if (MyData[i].GetType() == typeof(bool))  
    {  
        bool n = Convert.ToBoolean(MyData[i]);  
        Console.WriteLine(n);  
    }  
    else if (MyData[i].GetType() == typeof(DateTime))  
    {  
        DateTime n = Convert.ToDateTime(MyData[i]);  
        Console.WriteLine(n);  
    }  
}  
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/185. OOP - Typeof/TypeofExample/bin/Debug/TypeofExample.EXE
100
150
100.2398
hyderabad
True
9/16/2016 4:46:22 PM
```

The "System.String" class (or) String Handling

The “System.String” class

- In .NET, every string literal (a group of characters with double quotes) is automatically treated as an object for "System.String" class.
- The “System.String” class provides a set of properties and methods to manipulate strings.

Properties of “System.String” class:

- 1) `int Length`
 - It returns the no. of characters in the string.

Methods of “System.String” class:

- 1) `string ToUpper()`
 - It returns the string in upper case.
- 2) `string ToLower()`
 - It returns the string in lower case.
- 3) `char [int index]`
 - It returns the character at specified index.
- 4) `string Substring(int startIndex)`
 - It returns the part of the string, starting from specified index, up to end of the string.

5) `string Substring(int startIndex, int length)`

- It returns the part of the string, starting from specified index, the specified no. of characters.

6) `string Remove(int index)`

- It removes the part of the string, starting from specified index, up to end of the string, and returns the remaining string.

7) `string Remove(int index, int length)`

- It removes the part of the string, starting from specified index, up to specified length, and returns the remaining string.

8) `string Insert(int index, string value)`

- It inserts the given string value in the main string at the specified index and returns the result string.

9) `bool Equals(string value)`

- It compares the main string and string value, whether those are equal or not. It returns true if the both are equal. It returns false if both are not equal.

10) `bool Equals(string value, StringComparison.OrdinalIgnoreCase)`

- Same as “Equals” method, but it ignores the case (upper case / lower case).

11) `bool StartsWith(string value)`

- It returns true if the given value exists at the beginning of string. It returns false if the give value exists at the beginning of string.

12) `bool EndsWith(string value)`

- It returns true if the given value exists at the ending of string. It returns false if the give value not exists at the ending of string.

13) `bool Contains(string value)`

- It returns true if the given value exists in the string. It returns false if the given value exists in the string.

14) `int IndexOf(string value)`

- It searches and returns the index of first character of given value in the string. It returns -1 if is not found.

15) `int IndexOf(string value, int startIndex)`

- It is same as previous method, but searching starts from specified startIndex.

16) `int LastIndexOf(string value, int startIndex)`

- It is same as IndexOf() method, but searching starts from end of the string. (searching will be done from right-to-left).

17) `string Replace(string oldValue, string newValue)`

- It replaces each occurrence of oldValue with newValue and returns it.

18) `char[] ToCharArray()`

- It converts and returns the string as a character array.

19) `string[] Split(char separator)`

- It splits the string into small strings on each occurrence of given separator and returns all small strings as a string array.

20) `string Trim()`

- It returns the un-necessary spaces at the beginning and ending of the string; and it returns the final string.

21) `static string Format(string format, object arg0, object arg1, ...)`

- It substitutes place holders with the arguments, and returns the final string. It is a static method.

Note: No string method modifies the existing string object. It creates a new string object, stores the result string in it and returns the same string object.

“System.String.Length” - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “LengthExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LengthExample”.
- Click on OK.

Program.cs

```
using System;

namespace LengthExample
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "Hyderabad";

            //get length (no. of characters in the string)
            int len = s.Length;

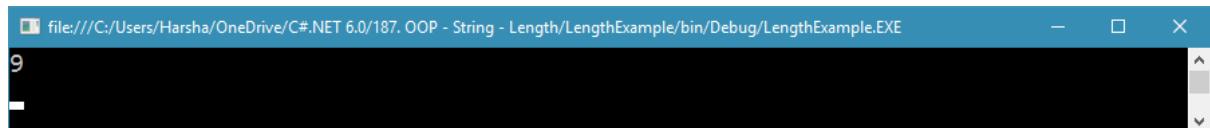
            //display length (no. of characters in the string)
            Console.WriteLine(len); //Output: 9

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
9
```

“System.String.ToUpper” - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ToUpperExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ToUpperExample”.
- Click on OK.

Program.cs

```
using System;

namespace ToUpperExample
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "Hyderabad";
        }
    }
}
```

```
//convert the string into upper case
string s2 = s.ToUpper();

//display
Console.WriteLine(s); //Output: Hyderabad
Console.WriteLine(s2); //Output: HYDERABAD

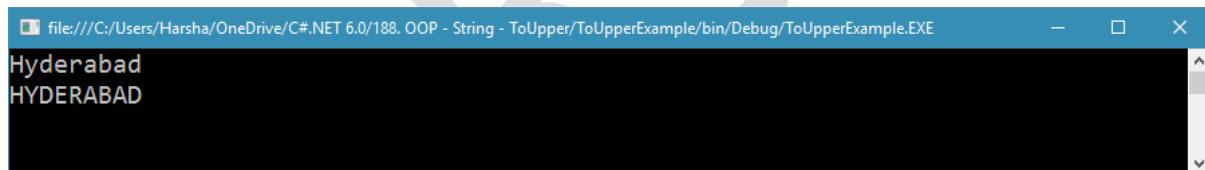
Console.ReadKey();
}

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.ToLower” - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ToLowerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ToLowerExample”.
- Click on OK.

Program.cs

```
using System;

namespace ToLowerExample
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "HYDERABAD";

            //convert into lower case
            string s2 = s.ToLower();

            //display
            Console.WriteLine(s); //Output: HYDERABAD
            Console.WriteLine(s2); //Output: hyderabad

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.String.GetChar" - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “GetCharExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “GetCharExample”.
- Click on OK.

Program.cs

```
using System;

namespace GetCharExample
{
    class Program
    {
```

```
static void Main()
{
    //create a string
    string s = "Hyderabad";

    //get the character at index no. "4".
    char ch = s[4];

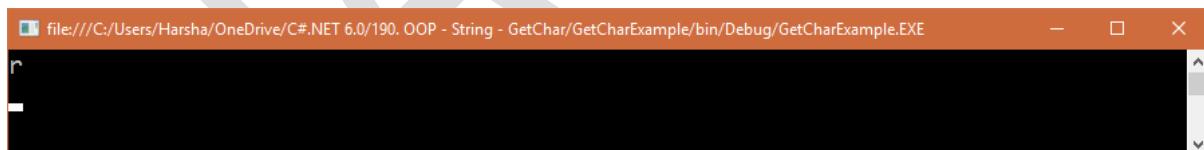
    //display
    Console.WriteLine(ch); //Output: r

    Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Substring” – Example 1

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SubstringExample1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SubstringExample1”.
- Click on OK.

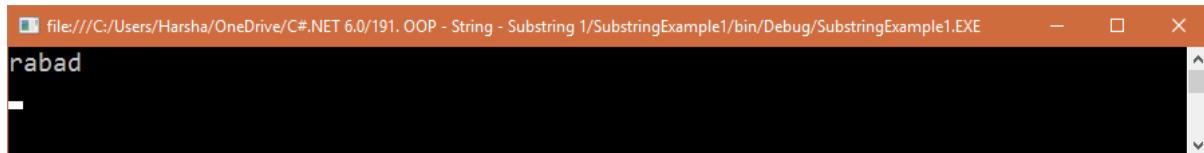
Program.cs

```
using System;
namespace SubstringExample1
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "Hyderabad";
            //get sub string (from 4th character - till end of the string)
            string s2 = s.Substring(4);
            //display
            Console.WriteLine(s2); //Output: rabad
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Substring” – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SubstringExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SubstringExample2”.
- Click on OK.

Program.cs

```
using System;
```

```
namespace SubstringExample2
```

```
{
```

```
    class Program
```

```

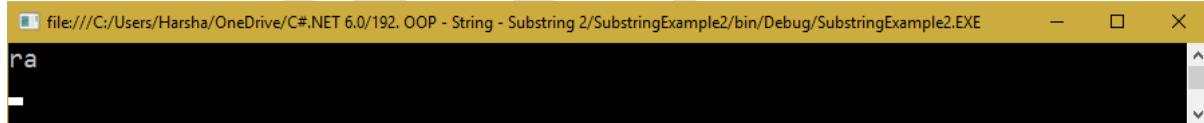
{
    static void Main()
    {
        //create a string
        string s = "hyderabad";
        //get sub string (from 4th character - only 2 characters length)
        string s2 = s.Substring(4, 2);
        //display
        Console.WriteLine(s2); //Output: ra
        Console.ReadKey();
    }
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Remove” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “RemoveExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RemoveExample”.
- Click on OK.

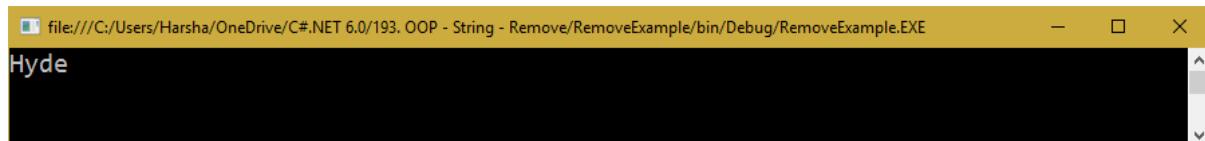
Program.cs

```
using System;
namespace RemoveExample
{
    class Program
    {
        static void Main()
        {
            //create string & integer
            string s = "Hyderabad";
            int ind = 4;
            //remove part of the string, starting from index "4" - till the end of
            //string & return the remaining part of the string.
            string s2 = s.Remove(ind);
            //display
            Console.WriteLine(s2); //Output: Hyde
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.String.Remove" – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Console Application".
- Type the project name as "RemoveExample2".
- Type the location as "C:\CSharp".
- Type the solution name as "RemoveExample2".
- Click on OK.

Program.cs

```
using System;
```

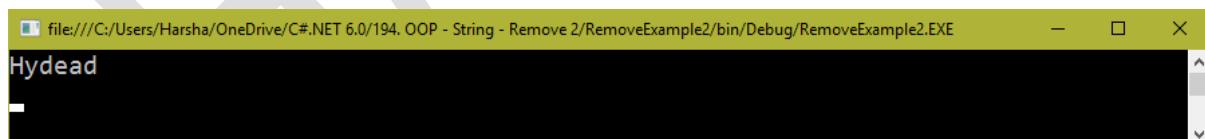
```
namespace RemoveExample2
{
    class Program
    {
        static void Main()
```

```
{  
    //create string & integers  
    string s = "Hyderabad";  
    int ind = 4;  
    int len = 3;  
  
    //remove part of the string, starting from index "4" - upto "3"  
    //characters length & get the remaining part of the string  
    string s2 = s.Remove(ind, len);  
  
    //display  
    Console.WriteLine(s2); //Output: Hydead  
  
    Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Insert” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “InsertExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InsertExample”.
- Click on OK.

Program.cs

```
using System;

namespace InsertExample
{
    class Program
    {
        static void Main()
        {
            //create strings & integer
            string s = "Hyderabad";
            string s2 = "abc";
            int ind = 4;

            //insert "abc" in "Hyderabad" at index no. 4
            string s3 = s.Insert(ind, s2);

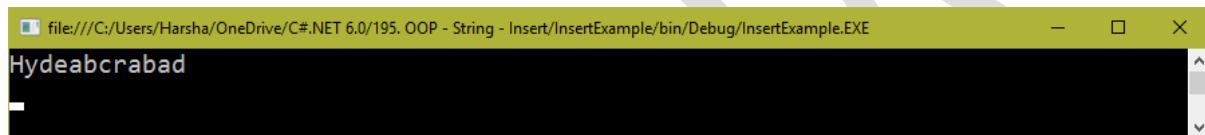
            //display
            Console.WriteLine(s3); //Output: Hydeabcrabad
        }
    }
}
```

```
    Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.String.Equals" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “EqualsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EqualsExample”.
- Click on OK.

Program.cs

```
using System;

namespace EqualsExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s1 = "Hyderabad";
            string s2 = "Hyderabad";

            //check whether s2 is equal to s1 or not
            bool b = s1.Equals(s2);

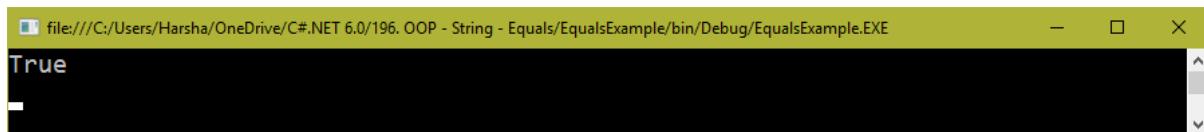
            //display
            Console.WriteLine(b); //Output: True

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a terminal window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/196. OOP - String - Equals/EqualsExample/bin/Debug/EqualsExample.EXE". The window contains the text "True" followed by a blank line.

“System.StringComparison.OrdinalIgnoreCase” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IgnoreCaseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IgnoreCaseExample”.
- Click on OK.

Program.cs

```
using System;

namespace IgnoreCaseExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s1 = "Hyderabad";
            string s2 = "hyderabad";

            //check whether s2 is equal to s1 or not, ignoring case
            bool b = s1.Equals(s2, StringComparison.OrdinalIgnoreCase);

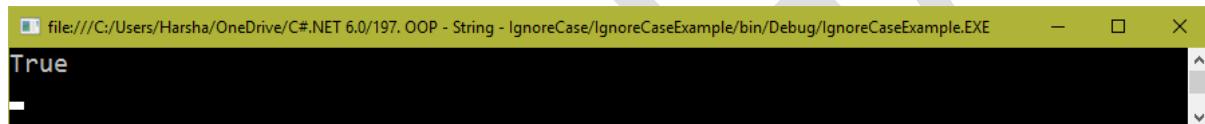
            //display
            Console.WriteLine(b); //Output: True
        }
    }
}
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.StartsWith” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StartsWithExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StartsWithExample”.
- Click on OK.

Program.cs

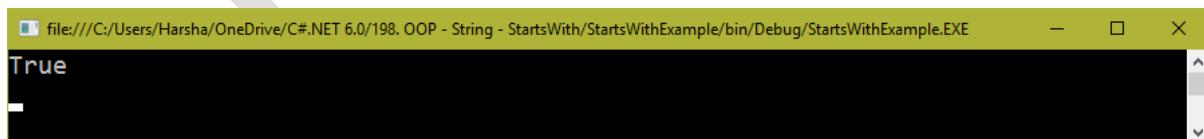
```
using System;

namespace StartsWithExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "hyderabad";
            string s2 = "hyd";
            //check whether "hyderabad" starting with "hyd" or not
            bool b = s.StartsWith(s2);
            Console.WriteLine(b); //Output: True
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.EndsWith” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “EndsWithExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EndsWithExample”.
- Click on OK.

Program.cs

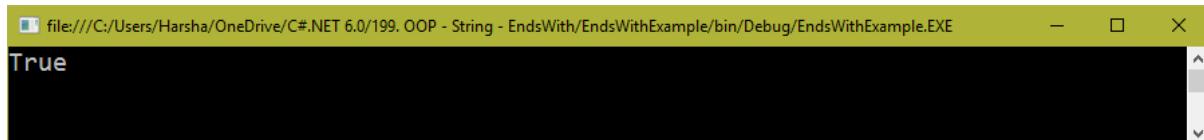
```
using System;

namespace EndsWithExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "hyderabad";
            string s2 = "bad";
            //check whether "hyderabad" ending with "bad" or not
            bool b = s.EndsWith(s2);
            Console.WriteLine(b); //Output: True
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Contains” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ContainsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ContainsExample”.
- Click on OK.

Program.cs

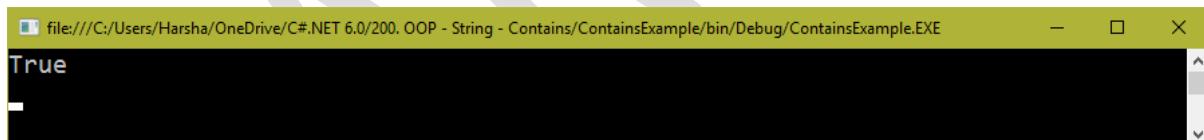
```
using System;
namespace ContainsExample
{
    class Program
```

```
{  
    static void Main()  
    {  
        //create strings  
        string s = "Hyderabad";  
        string s2 = "dera";  
        //check whether "Hyderabad" contains "dera" or not  
        bool b = s.Contains(s2);  
        //display  
        Console.WriteLine(b); //Output: True  
        Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.IndexOf” – Example 1

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IndexOfExample1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IndexOfExample1”.
- Click on OK.

Program.cs

```
using System;

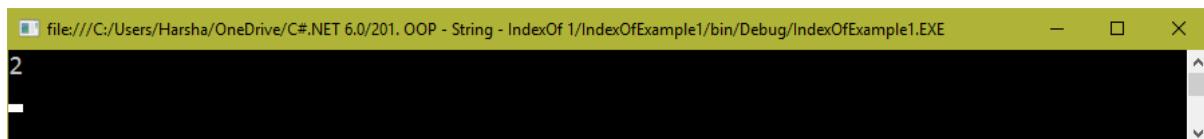
namespace IndexOfExample1
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "Hyderabad";
            string s2 = "dera";
            //get the index of "dera" in "Hyderabad"
            int ind = s.IndexOf(s2);
            //display
            Console.WriteLine(ind); //2
            Console.ReadKey();
        }
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.IndexOf” – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IndexOfExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IndexOfExample2”.
- Click on OK.

Program.cs

using System;

```

namespace IndexOfExample2
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "hyderabad";
            string s2 = "a";

            //get the index of "a" in "hyderabad" - first occurrence
            int ind1 = s.IndexOf(s2);
            Console.WriteLine(ind1); //5

            //get the index of "a" in "hyderabad" - second occurrence
            int ind2 = s.IndexOf(s2, ind1+1);
            Console.WriteLine(ind2); //7

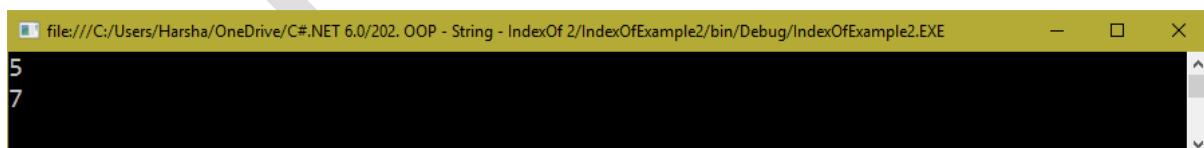
            Console.ReadKey();
        }
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/202. OOP - String - IndexOf 2/IndexOfExample2/bin/Debug/IndexOfExample2.EXE
5
7

```

“System.String.IndexOf” – Example 3

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IndexOfExample3”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IndexOfExample3”.
- Click on OK.

Program.cs

```
using System;

namespace IndexOfExample3
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "Hyderabad";
            string s2 = "xyz";

            //get the index of "xyz" in "Hyderabad". As it is not found, it returns -1
            int ind = s.IndexOf(s2);

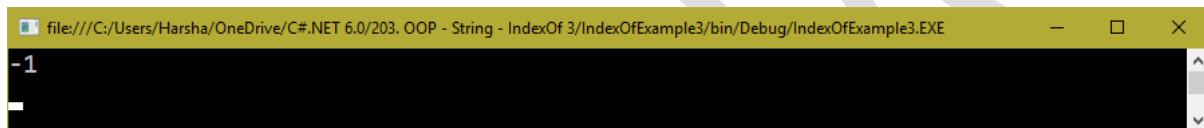
            //display
            Console.WriteLine(ind); //Output: -1
        }
    }
}
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.LastIndexOf” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “LastIndexOfExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LastIndexOfExample”.
- Click on OK.

Program.cs

```
using System;

namespace LastIndexOfExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "Hyderabad";
            string s2 = "a";

            //get the last index of "a" in "Hyderabad" - Last occurrence
            int ind = s.LastIndexOf(s2);

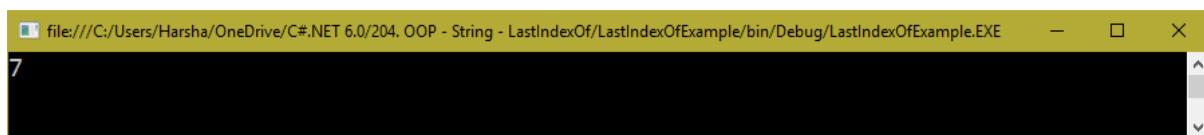
            //display
            Console.WriteLine(ind); //7

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Replace” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReplaceExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ReplaceExample”.
- Click on OK.

Program.cs

```
using System;

namespace ReplaceExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "Apple Smart Phones";
            string s2 = "Apple";
            string s3 = "Samsung";

            //replace "Apple" with "Samsung"
            string s4 = s.Replace(s2, s3);

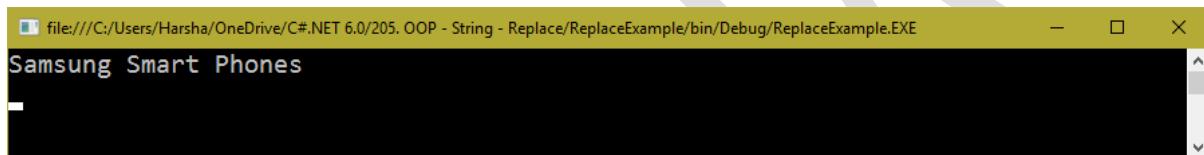
            Console.WriteLine(s4); //Output: Samsung Smart Phones
        }
    }
}
```

```
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.ToCharArray” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ToCharArrayExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ToCharArrayExample”.
- Click on OK.

Program.cs

```
using System;

namespace ToCharArrayExample
{
    class Program
    {
        static void Main()
        {
            //create string
            string s = "hyderabad";

            //convert "string" into "character array"
            char[ ] ch = s.ToCharArray();

            //read each character from the "character array" using for loop
            for (int i = 0; i < ch.Length; i++)
            {
                Console.WriteLine(ch[i]);
            }

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/206. OOP - String - ToCharArray/ToCharArrayExample/bin/Debug/ToCharArrayExample.EXE
```

```
hyderabad
```

Converting CharArray to String - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CharArrayToStringExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CharArrayToStringExample”.
- Click on OK.

Program.cs

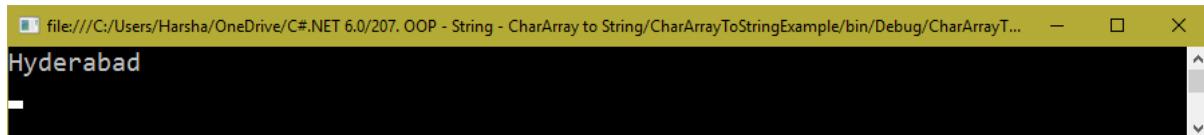
```
using System;

namespace CharArrayToStringExample
{
    class Program
    {
        static void Main()
        {
            //create character array
            char[] ch = new char[9] { 'H', 'y', 'd', 'e', 'r', 'a', 'b', 'a', 'd' };
            //convert character array to string
            string s = new String(ch);
            //display string
            Console.WriteLine(s); //Output: Hyderabad
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/207. OOP - String - CharArray to String/CharArrayToStringExample/bin/Debug/CharArrayT...
Hyderabad
-
```

“System.String.Split” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SplitExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SplitExample”.
- Click on OK.

Program.cs

```
using System;
```

```
namespace SplitExample
```

```

{
    class Program
    {
        static void Main()
        {
            //create string and character
            string s = "Hyderabad is one of the cities in India";
            char ch = ' ';

            //convert the "string" into "string array" at each occurrence of " "
            (space)
            string[] s2 = s.Split(ch);

            //read and display each element from the string array
            for (int i = 0; i < s2.Length; i++)
            {
                Console.WriteLine(s2[i]);
            }
            Console.ReadKey();
        }
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/208. OOP - String - Split/SplitExample/bin/Debug/SplitExample.EXE
Hyderabad
is
one
of
the
cities
in
India
-
```

“System.String.Trim” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “TrimExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TrimExample”.
- Click on OK.

Program.cs

```
using System;

namespace TrimExample
{
    class Program
    {
        static void Main()
        {
            //create string
            string s = " Hyderabad ";
            //get no. of characters
            int len1 = s.Length;
            Console.WriteLine(s); //Output: " Hyderabad "
            Console.WriteLine(len1); //Output: 16
            Console.WriteLine();
        }
    }
}
```

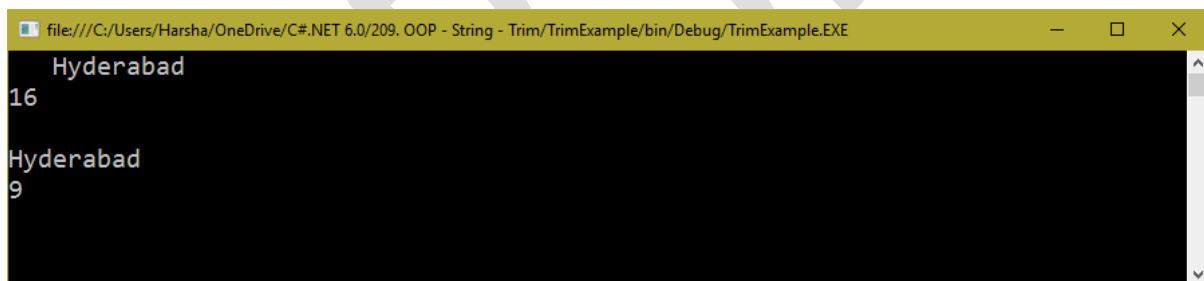
```
//trim the string (remove extra spaces at "most left side" and "most
right side")
string s2 = s.Trim();
int len2 = s2.Length;
Console.WriteLine(s2); //Output: "Hyderabad"
Console.WriteLine(len2); //Output: 9

Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Format” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “FormatExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormatExample”.
- Click on OK.

Program.cs

```
using System;

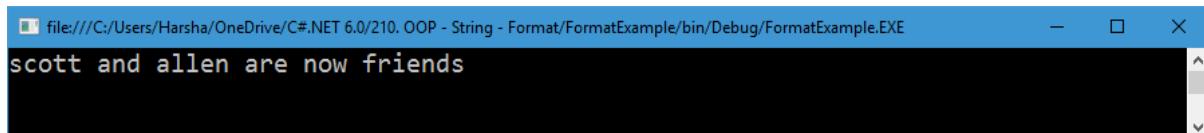
namespace FormatExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s1 = "scott";
            string s2 = "allen";
            string original = "{0} and {1} are now friends";
            //Substitute "scott" in {0} place and "allen" in {1} place.
            string final = string.Format(original, s1, s2);
            Console.WriteLine(final); //Output: scott and allen are now friends.
            Console.ReadKey();
        }
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



“System.String.Reverse” – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ReverseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ReverseExample”.
- Click on OK.

Program.cs

```
using System;
```

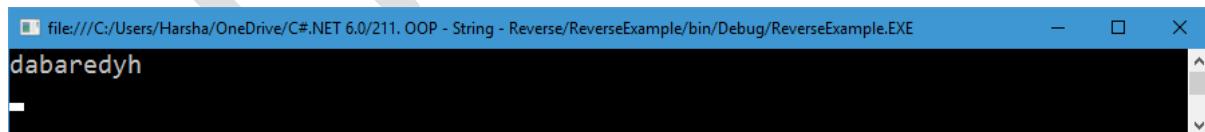
```
namespace ReverseExample
```

```
{  
    class Program  
    {  
        static void Main()  
        {  
            //create strings  
            string s = "hyderabad";  
            string s2 = "";  
  
            //read each character using for loop and append it to a new string  
            for (int i = s.Length - 1; i >= 0; i--)  
            {  
                s2 = s2 + s[i];  
            }  
  
            Console.WriteLine(s2); //dabaredyh  
            Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – WordsCount - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “WordsCountExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WordsCountExample”.
- Click on OK.

Program.cs

```
using System;

namespace WordsCountExample
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "She sells sea shells on the sea shore";

            //create counter variable
            int c = 0;

            //read each character and check it whether the character is " " (space)
            //or not. If it is space, increment the counter variable
            for (int i = 0; i < s.Length; i++)
            {
                if (s[i] == ' ')

```

```
    c++;
}

//increment the counter variable
if (s.Length >=1)
    c++;

Console.WriteLine(c); //Output: 8

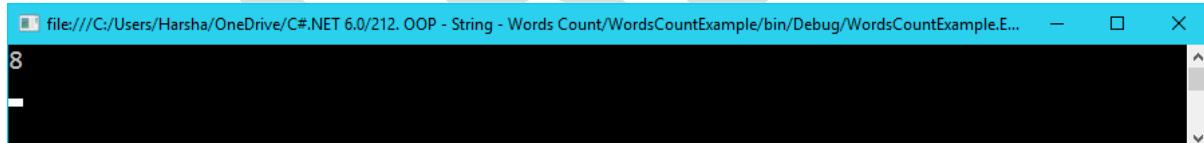
Console.ReadKey();
}

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – Character Occurrence Count - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CharacterOccurrenceCountExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CharacterOccurrenceCountExample”.
- Click on OK.

Program.cs

```
using System;

namespace CharacterOccurrenceCountExample
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "She sells sea shells on the sea shore";

            //create a character
            char ch = 's';

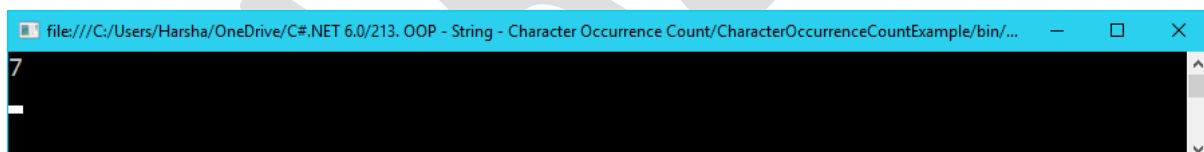
            //create a counter variable
            int c = 0;
```

```
//read each character and check whether the character matches or  
not. If matches, increment the counter variable  
for (int i = 0; i < s.Length; i++)  
{  
    if (s[i] == ch)  
        c++;  
}  
  
Console.WriteLine(c); //Output: 7  
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – Alphabetic Count - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “AlphabetsCountExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AlphabetsCountExample”.
- Click on OK.

Program.cs

```
using System;

namespace AlphabetsCountExample
{
    class Program
    {
        static void Main()
        {
            //create a string
            string s = "She sells sea shells on the sea shore";

            //create a counter variable
            int c = 0;

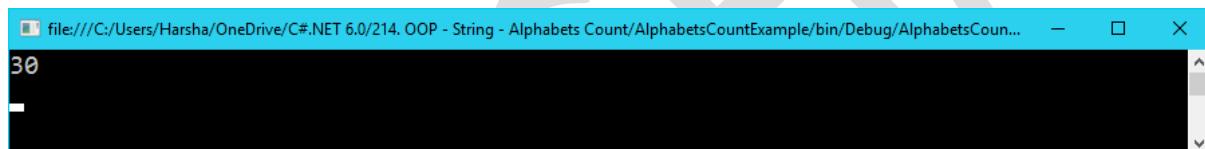
            //read each character and check whether it is alphabet or not, using
            //ASCII code
            for (int i = 0; i < s.Length; i++)
            {
                if ((s[i] >= 65 && s[i] <= 90) || (s[i] >= 97 && s[i] <= 122))
```

```
    C++;  
}  
  
Console.WriteLine(c); //Output: 30  
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – Word Occurrence Count - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “WordOccurrenceCountExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WordOccurrenceCountExample”.
- Click on OK.

Program.cs

```
using System;

namespace WordOccurrenceCountExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "She sells sea shells on the sea shore";
            string s2 = "sea";

            //create a counter variable
            int c = 0;

            //read each character
            for (int i = 0; i < s.Length; i++)
            {
```

```

//check whether the character matches with the first character of
"s2".
if (s[i] == s2[0] && (i + s2.Length) <= s.Length)
{
    bool flag = true;

    //check for remaining characters of "s2".
    for (int j = i, k = 0; k < s2.Length; j++, k++)
    {
        if (s[j] != s2[k])
            flag = false;
    }

    //if all the characters are matched, increment the counter variable
    if (flag == true)
        c++;
}
}

Console.WriteLine(c); //Output: 2

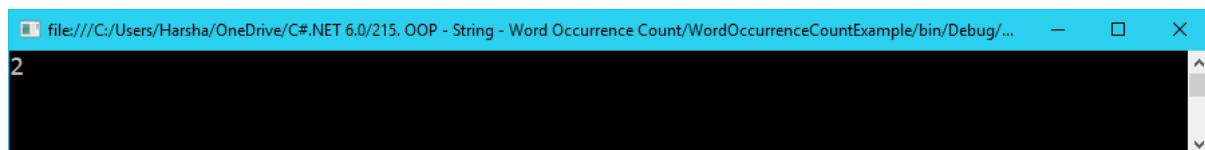
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – Title Case - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “TitleCaseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TitleCaseExample”.
- Click on OK.

Program.cs

```
using System;

namespace TitleCaseExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s = "she sells sea shells on the sea shore";
            string s2 = "";
            char ch = ' ';

            //split the string into "string array" at each occurrence of space.
            string[] words = s.Split(ch);

            //converting into "Title Case" (Every word starts with capital letter).
            for (int i = 0; i < words.Length; i++)
            {

```

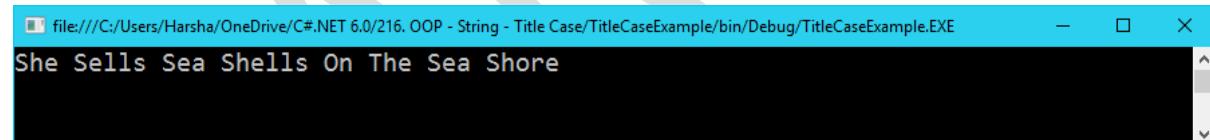
```
s2 += words[i][0].ToString().ToUpper() +  
words[i].Substring(1).ToLower() + " ";  
}  
  
//remove unnecessary spaces  
s2 = s2.Trim();  
  
Console.WriteLine(s2); //Output: She Sells Sea Shells On The Sea  
Shore
```

```
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – Currency into words - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CurrencyIntoWordsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CurrencyIntoWordsExample”.
- Click on OK.

Program.cs

```
using System;
using System.Text;

namespace CurrencyIntoWordsExample
{
    class Program
    {
        static void Main()
        {
            //actual number that is to be converted into currency
            int number = 1234;

            //create string
            string result;

            //check whether the number is zero
            if (number == 0)
            {
                result = "Zero";
            }
        }
    }
}
```

```

    goto end;
}

//create integer array
int[ ] num = new int[4];

int first = 0;
int u, h, t;

//create stringbuilder
StringBuilder sb = new StringBuilder();

//check whether the number is negetive
if (number < 0)
{
    sb.Append("Minus ");
    number = -number;
}

//create string arrays
string[] words0 = {"", "One ", "Two ", "Three ", "Four ", "Five ", "Six ",
"Seven ", "Eight ", "Nine "};
string[] words1 = {"Ten ", "Eleven ", "Twelve ", "Thirteen ", "Fourteen ",
"Fifteen ", "Sixteen ", "Seventeen ", "Eighteen ", "Nineteen "};
string[] words2 = {"Twenty ", "Thirty ", "Forty ", "Fifty ", "Sixty ",
"Seventy ", "Eighty ", "Ninety "};
string[] words3 = { "Thousand ", "Lakh ", "Crore "};

num[0] = number % 1000; // units
num[1] = number / 1000;
num[2] = number / 100000;
num[1] = num[1] - 100 * num[2]; // thousands
num[3] = number / 10000000; // crores
num[2] = num[2] - 100 * num[3]; // lakhs

for (int i = 3; i > 0; i--)
{

```

```

if (num[i] != 0)
{
    first = i;
    break;
}
}

for (int i = first; i >= 0; i--)
{
    if (num[i] == 0) continue;
    u = num[i] % 10; // ones
    t = num[i] / 10;
    h = num[i] / 100; // hundreds
    t = t - 10 * h; // tens
    if (h > 0) sb.Append(words0[h] + "Hundred ");
    if (u > 0 || t > 0)
    {
        if ((h > 0 || i == 0) && (number.ToString().Length > 3))
sb.Append("and ");
        if (t == 0)
            sb.Append(words0[u]);
        else if (t == 1)
            sb.Append(words1[u]);
        else
            sb.Append(words2[t - 2] + words0[u]);
    }
    if (i != 0) sb.Append(words3[i - 1]);
}
result = sb.ToString().TrimEnd();
end:

Console.WriteLine(number);
Console.WriteLine(result);

Console.ReadKey();
}
}

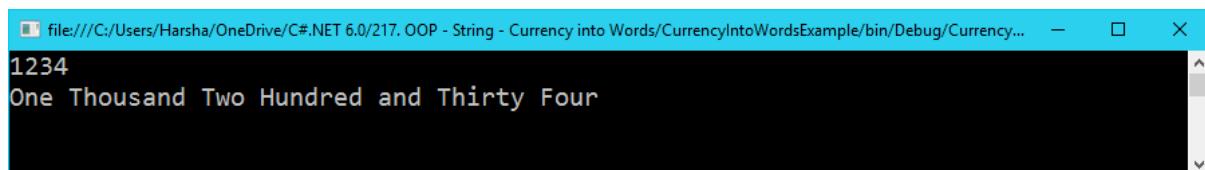
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



String – Multiple Concatenations - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MultipleConcatenationsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultipleConcatenationsExample”.
- Click on OK.

Program.cs

```
using System;

namespace MultipleConcatenationsExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s1, s2, s3, s4, result;
            s1 = "Hello "; //string object 1
            s2 = "how "; //string object 2
            s3 = "are "; //string object 3
            s4 = "you"; //string object 4

            //concatenations
            result = s1;
            result = result + s2; //string object 5
            result = result + s3; //string object 6
```

```
result = result + s4; //string object 7
```

```
Console.WriteLine(result); //Output: Hello how are you
```

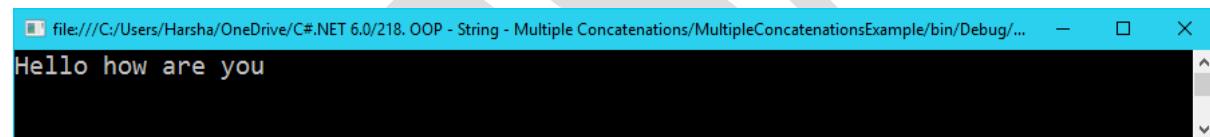
```
Console.ReadKey();
}
}
}
```

```
//Total no. of objects created in the program = 7
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.Text.StringBuilder" class

System.Text.StringBuilder

- “StringBuilder” is a pre-defined class in “System.Text” namespace.
- This class’s object represents an appendable string.
- The string builder can be appendable any no. of times, without creating a separate object.
- String concatenation always creates a new object for “System.String” class; but string builder allows the value to be appended any no. of times. So string builder saves memory when compared with string concatenation, in case of many appendings.

Steps for development of StringBuilder:

- Import the namespace:
 - ❖ `using System.Text;`
- Create a reference variable:
 - ❖ `StringBuilder sb;`
- Create an object for StringBuilder class:
 - ❖ `sb = new StringBuilder();`
- Append string:
 - ❖ `sb.Append(string value);`
- Convert the stringbuilder to string:
 - ❖ `sb.ToString();`

The "System.Text.StringBuilder" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “StringBuilderExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StringBuilderExample”.
- Click on OK.

Program.cs

```
using System;
using System.Text;

namespace StringBuilderExample
{
    class Program
    {
        static void Main()
        {
            //create strings
            string s1, s2, s3, s4;
            s1 = "Hello "; //string object 1
            s2 = "how "; //string object 2
            s3 = "are "; //string object 3
            s4 = "you"; //string object 4

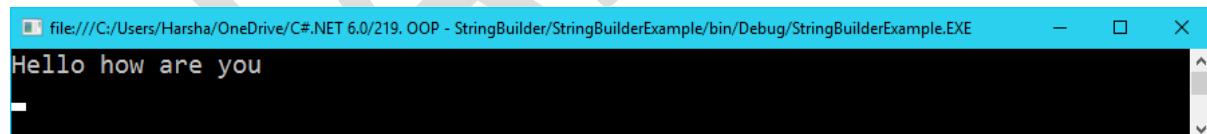
            //create reference variable
            StringBuilder result;
```

```
//create object for StringBuilder class  
result = new StringBuilder();  
  
//append strings to the stringbuilder  
result.Append(s1);  
result.Append(s2);  
result.Append(s3);  
result.Append(s4);  
  
Console.WriteLine(result.ToString()); //Output: Hello how are you  
Console.ReadKey();  
}  
}  
}  
  
//Total no. of objects created in the program = 4
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime" structure (or) Date & Time Handling

The “System.DateTime” structure

- .NET provides a pre-defined structure called “DateTime”, to store and manipulate date & time.
- “DateTime” is a structure; “System” is a namespace.
- The “DateTime” structure provides several properties and methods to perform manipulations on date and time.

Properties of “System.DateTime” structure:

- 1) DateTime Now
 - It returns the current system date and time.
- 2) int Day
 - It returns the current day. (1 to 31).
- 3) int Month
 - It returns the current month. (1 to 12).
- 4) int Year
 - It returns the current year. (1 to 12).
- 5) int Hour
 - It returns the current hour.
- 6) int Minute

- It returns the current minute.
- 7) int Second
- It returns the current second.
- 8) int Millisecond
- It returns the current millisecond.
- 9) string DayOfWeek.ToString()
- It returns the day name of the week.
- 10) int DayOfYear
- It returns the current day's index in the current year.
-
- ## Methods of “System.DateTime” structure:
-
- 11) string ToShortDateString()
- It returns the date in the following format: M/d/yyyy
- 12) string ToLongDateString()
- It returns the date in the following format: dddd, MMMM dd, yyyy
- 13) string ToShortTimeString()
- It returns the time in the following format: h:m tt
- 14) string ToLongTimeString()
- It returns the time in the following format: h:mm:ss tt

15) `string ToString("format")`

- It returns the date / time in the desired format.

16) `TimeSpan Subtract(DateTime dt2)`

- It subtracts the main date with given dt2, and returns the difference dates in the form of TimeSpan.

17) `DateTime AddDays(int n)`

- It adds the 'n' no. of days to the main date and returns the result date.

The "System.DateTime" – First Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#". Select "Console Application".
- Type the project name as "DateTimeFirstExample".
- Type the location as "C:\CSharp".
- Type the solution name as "DateTimeFirstExample".
- Click on OK.

Program.cs

`using System;`

`namespace DateTimeFirstExample`

```
{  
    class Program  
    {  
        static void Main()  
        {  
            //create a variable of DateTime data type  
            DateTime d;  
            //set value  
            d = Convert.ToDateTime("2/10/2016 5:30 PM");  
            //get value  
            Console.WriteLine(d);  
            Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime.Now" –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “NowExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NowExample”.
- Click on OK.

Program.cs

```
using System;

namespace NowExample
{
    class Program
    {
        static void Main()
        {
            //create a variable of DateTime data type
            DateTime d;
            //set system date and time into the variable
            d = DateTime.Now;
            //get value
            Console.WriteLine(d);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime" – Inner Values – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “InnerValuesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InnerValuesExample”.
- Click on OK.

Program.cs

```
using System;

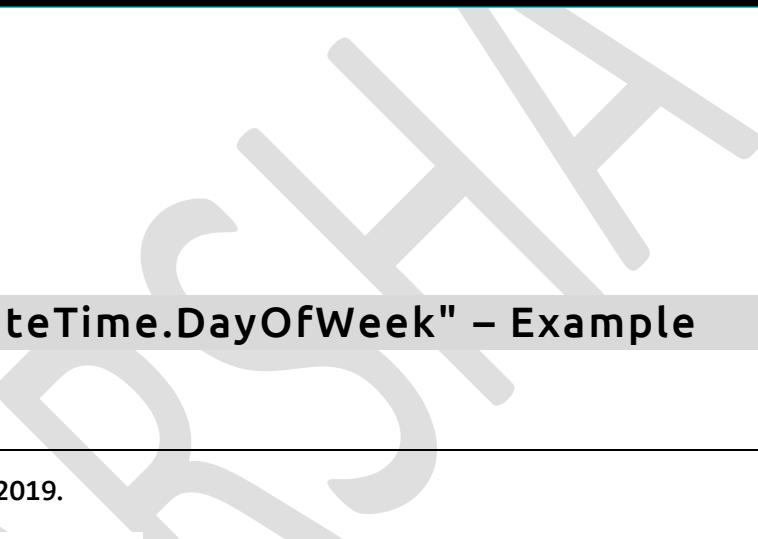
namespace InnerValuesExample
{
    class Program
    {
        static void Main()
```

```
{  
    //create a variable of DateTime data type  
    DateTime d;  
  
    //set system date and time into the variable  
    d = DateTime.Now;  
  
    //get individual values  
    int dy = d.Day;  
    int m = d.Month;  
    int y = d.Year;  
    int h = d.Hour;  
    int mi = d.Minute;  
    int s = d.Second;  
    int ms = d.Millisecond;  
  
    //display values  
    Console.WriteLine("Day: " + dy);  
    Console.WriteLine("Month: " + m);  
    Console.WriteLine("Year: " + y);  
    Console.WriteLine();  
    Console.WriteLine("Hour: " + h);  
    Console.WriteLine("Minute:" + mi);  
    Console.WriteLine("Second: " + s);  
    Console.WriteLine();  
    Console.WriteLine("Milli Second: " + ms);  
    Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/222. OOP - DateTime - Inner Values/InnerValuesExample/bin/Debug/InnerValuesExample.EXE – X
Day: 16
Month: 9
Year: 2016

Hour: 20
Minute:7
Second: 26

Milli Second: 712
```

The "System.DateTime.DayOfWeek" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “DayOfWeekExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DayOfWeekExample”.
- Click on OK.

Program.cs

```
using System;
```

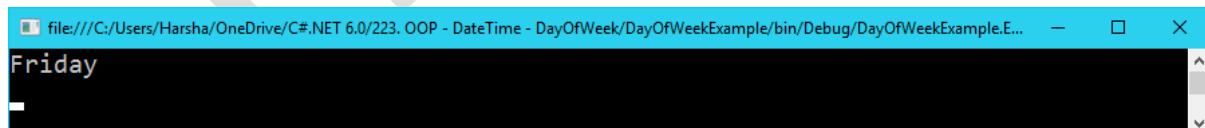
```
namespace DayOfWeekExample
```

```
{  
    class Program  
    {  
        static void Main()  
        {  
            //create variable of DateTime data type  
            DateTime d;  
  
            //set system date and time into the variable  
            d = DateTime.Now;  
  
            //get name of the day of week  
            string str = d.DayOfWeek.ToString();  
            Console.WriteLine(str);  
  
            Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime.DayOfYear" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DayOfYearExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DayOfYearExample”.
- Click on OK.

Program.cs

```
using System;

namespace DayOfYearExample
{
    class Program
    {
        static void Main()
        {
            //create variable of DateTime data type
            DateTime d;

            //set system date and time into the variable
            d = DateTime.Now;

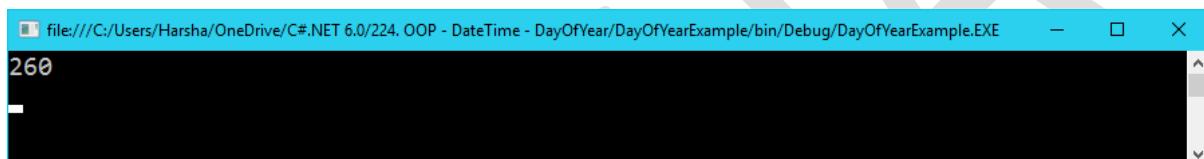
            //get no. of the day in the current year (1 to 365)
            int n = d.DayOfYear;
            Console.WriteLine(n);
        }
    }
}
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime.ToShortDateString" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “ShortDateExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ShortDateExample”.
- Click on OK.

Program.cs

```
using System;

namespace ShortDateExample
{
    class Program
    {
        static void Main()
        {
            //create variable of DateTime data type
            DateTime d;

            //set system date and time into the variable
            d = DateTime.Now;

            //get short date
            string str = d.ToShortDateString();
            Console.WriteLine(str);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime. ToLongDateString" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “LongDateExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LongDateExample”.
- Click on OK.

Program.cs

```
using System;

namespace LongDateExample
{
    class Program
    {
        static void Main()
        {
            //create a variable of DateTime data type
            DateTime d;

            //set system date and time into the variable
            d = DateTime.Now;

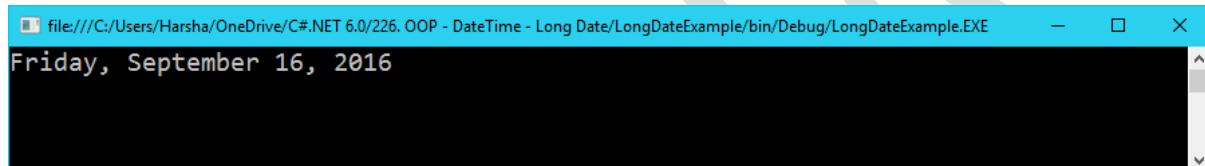
            //get long date
            string str = d.ToString("yyyy-MM-dd HH:mm:ss");
            Console.WriteLine(str);
        }
    }
}
```

```
    Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime. ToShortTimeString" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ShortTimeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ShortTimeExample”.
- Click on OK.

Program.cs

```
using System;

namespace ShortTimeExample
{
    class Program
    {
        static void Main()
        {
            //create a variable of DateTime data type
            DateTime d;

            //set system date and time into the variable
            d = DateTime.Now;

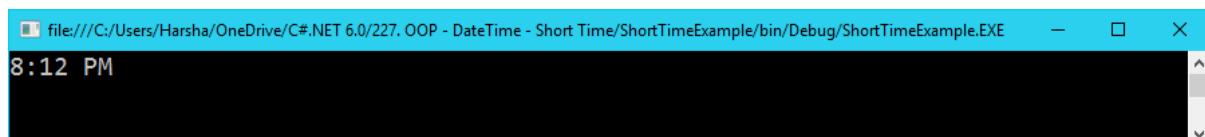
            //get short time
            string str = d.ToShortTimeString();
            Console.WriteLine(str);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime. ToLongTimeString" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “LongTimeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LongTimeExample”.
- Click on OK.

Program.cs

```
using System;

namespace LongTimeExample
{
    class Program
    {
        static void Main()
        {
            //create a variable of DateTime data type
            DateTime d;

            //set system date and time into the variable
            d = DateTime.Now;

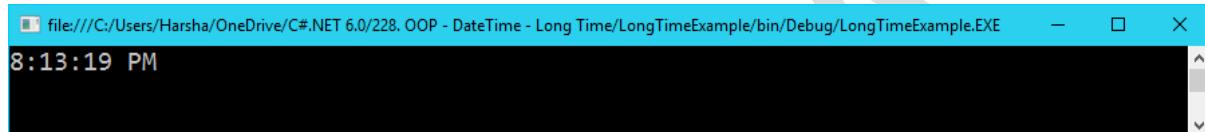
            //get long time
            string str = d.ToString("yyyy-MM-dd HH:mm:ss");
            Console.WriteLine(str);
        }
    }
}
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.DateTime" – Custom Formats – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CustomFormatsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CustomFormatsExample”.

- Click on OK.

Program.cs

```
using System;

namespace CustomFormatsExample
{
    class Program
    {
        static void Main()
        {
            //create a variable of DateTime data type
            DateTime d;

            //set system date and time into the variable
            d = DateTime.Now;

            //get custom formats
            string str1 = d.ToString("d/M/yyyy");
            string str2 = d.ToString("yyyy/M/d");
            string str3 = d.ToString("h:m:s");

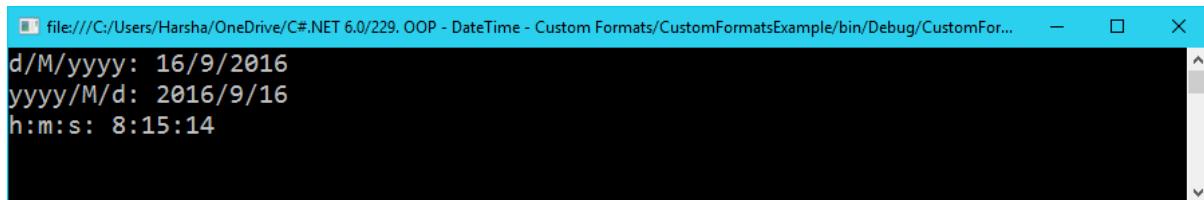
            //display values
            Console.WriteLine("d/M/yyyy: " + str1);
            Console.WriteLine("yyyy/M/d: " + str2);
            Console.WriteLine("h:m:s: " + str3);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/229. OOP - DateTime - Custom Formats/CustomFormatsExample/bin/Debug/CustomFor...
d/M/yyyy: 16/9/2016
yyyy/M/d: 2016/9/16
h:m:s: 8:15:14
```

The "System.DateTime.Subtract" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SubtractExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SubtractExample”.
- Click on OK.

Program.cs

```
using System;

namespace SubtractExample
{
    class Program
    {
```

```
static void Main()
{
    //create variables of DateTime data type
    DateTime dt1, dt2;
    dt1 = DateTime.Now;
    dt2 = Convert.ToDateTime("7/8/2011"); ;

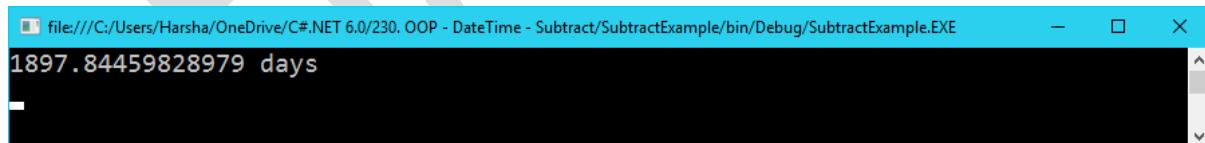
    //subtract dt2 from dt1 (get the no. of days between 7/8 /2011 till
    date)
    TimeSpan ts;
    ts = dt1.Subtract(dt2);

    //display the no. of days difference between dt1 and dt2 (dt1 - dt2)
    string msg = ts.TotalDays + " days";
    Console.WriteLine(msg);
    Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/230. OOP - DateTime - Subtract/SubtractExample/bin/Debug/SubtractExample.EXE". The main window displays the text "1897.84459828979 days". The window has standard minimize, maximize, and close buttons at the top right.

The "System.DateTime.AddDays" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”. Select “Console Application”.
- Type the project name as “AddDaysExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AddDaysExample”.
- Click on OK.

Program.cs

```
using System;
namespace AddDaysExample
{
    class Program
    {
        static void Main()
        {
            //create a variable of DateTime data type
            DateTime d;
            //set system date and time into the variable
            d = DateTime.Now;
            //add 10 days
            DateTime temp = d.AddDays(10);
            Console.WriteLine(temp);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/231. OOP - DateTime - Add Days/AddDaysExample/bin/Debug/AddDaysExample.EXE
9/26/2016 8:17:18 PM
```

The "System.Math" class

The “System.Math” class

- .NET provides a pre-defined class called “Math”, to store perform mathematical manipulations.
- “Math” is a class; “System” is a namespace.
- “Math” is a static class; so all of its members are “static members”.

Methods of “System.Math” class:

- 1) double Abs(double value)
 - It returns the absolute value (positive value) of given number.
- 2) double Floor(double value)
 - It returns the number without decimal part.
- 3) double Ceiling(double value)
 - It returns the next number without decimal part.
- 4) double Round(double value)
 - It returns the current number without decimal part, if the first digit of the decimal part is less than 5.
 - It returns the next number without decimal part, if the first digit of the decimal part is greater than or equal to 5.
- 5) double Max(double n1, double n2)

- It returns the big number.
- 6) double Min(double n1, double n2)
- It returns the small number.
- 7) double Pow(double n1, double n2)
- It returns n1 power n2.
- 8) double Sqrt(double n)
- It returns the square root of 'n'.

The "System.Math.Abs" –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AbsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AbsExample”.
- Click on OK.

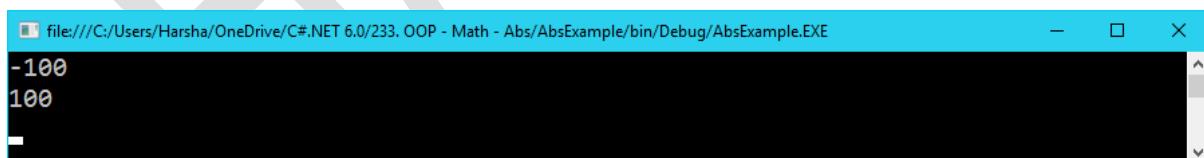
Program.cs

```
using System;
namespace AbsExample
{
    class Program
    {
        static void Main()
        {
            double n = -100;
            double result;
            //convert the negative value into positive value
            result = Math.Abs(n);
            Console.WriteLine(n);
            Console.WriteLine(result);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.Math.Floor" –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “FloorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FloorExample”.
- Click on OK.

Program.cs

```
using System;

namespace FloorExample
{
    class Program
    {
        static void Main()
        {
            double n = 100.3492;
            double result;
            //get the floor value (without decimal part)
            result = Math.Floor(n);
            Console.WriteLine(n);
            Console.WriteLine(result);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/234. OOP - Math - Floor/FloorExample/bin/Debug/FloorExample.EXE
100.3492
100
```

The "System.Math.Ceiling" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “CeilingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CeilingExample”.
- Click on OK.

Program.cs

```
using System;
namespace CeilingExample
{
    class Program
    {
```

```

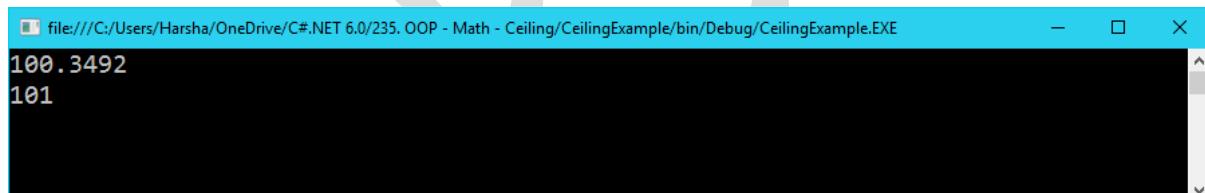
static void Main()
{
    double n = 100.3492;
    double result;
    //get the next integer value
    result = Math.Ceiling(n);
    Console.WriteLine(n);
    Console.WriteLine(result);
    Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window with the title bar "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/235. OOP - Math - Ceiling/CeilingExample/bin/Debug/CeilingExample.EXE". The window contains the following text:

```

100.3492
101

```

The "System.Math.Round" – Example 1

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “RoundExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RoundExample”.
- Click on OK.

Program.cs

```
using System;

namespace RoundExample
{
    class Program
    {
        static void Main()
        {
            double n = 100.3492;
            double result;

            //get the next number if the first digit in decimal part is greater than
            //or equal to "5"; get the previous number if the first digit in decimal part is
            //less than "5".
            result = Math.Round(n);

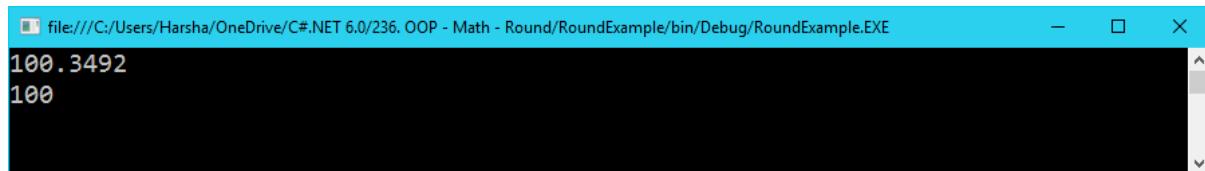
            Console.WriteLine(n);
            Console.WriteLine(result);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/236. OOP - Math - Round/RoundExample/bin/Debug/RoundExample.EXE
100.3492
100
```

The "System.Math.Round" – Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “RoundExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RoundExample2”.
- Click on OK.

Program.cs

```
using System;

namespace RoundExample2
{
    class Program
    {
        static void Main()
        {
            double n = 100.6492;
```

```

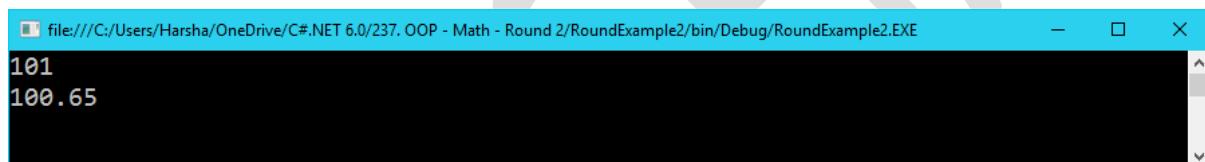
double result1, result2;
result1 = Math.Round(n);
result2 = Math.Round(n, 2);
Console.WriteLine(result1);
Console.WriteLine(result2);
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.Math.Max" –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MaxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MaxExample”.

- Click on OK.

Program.cs

```
using System;

namespace MaxExample
{
    class Program
    {
        static void Main()
        {
            double n1 = 100, n2 = 200;
            double result;
            //get the big value among n1 and n2
            result = Math.Max(n1, n2);
            Console.WriteLine(result);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.Math.Min" – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MinExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MinExample”.
- Click on OK.

Program.cs

```
using System;

namespace MinExample
{
    class Program
    {
        static void Main()
        {
            double n1 = 100, n2 = 200;
            double result;
            //get the smaller value among n1 and n2
            result = Math.Min(n1, n2);
            Console.WriteLine(result);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The "System.Math.Pow" –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “PowExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “PowExample”.
- Click on OK.

Program.cs

```
using System;

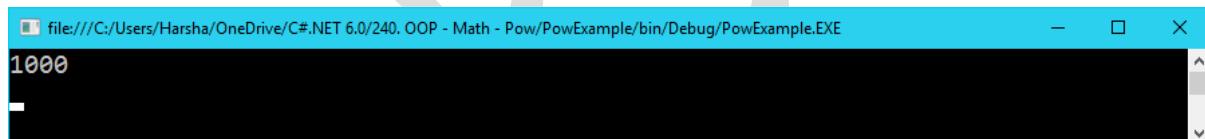
namespace PowExample
{
    class Program
```

```
{  
    static void Main()  
    {  
        double n1 = 10, n2 = 3;  
        double result;  
        //get the result of "10" power "3"  
        result = Math.Pow(n1, n2);  
        Console.WriteLine(result);  
        Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a terminal window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/240. OOP - Math - Pow/PowExample/bin/Debug/PowExample.EXE". The window shows the output of the program: "1000" followed by a new line character "-". The terminal has a blue header bar and a black body with white text.

The "System.Math.Sqrt" –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SqrtExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqrtExample”.
- Click on OK.

Program.cs

```
using System;

namespace SqrtExample
{
    class Program
    {
        static void Main()
        {
            double n = 10;
            double result;
            //get square root of 10
            result = Math.Sqrt(n);
            Console.WriteLine(result);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/241. OOP - Math - Sqrt/SqrtExample/bin/Debug/SqrtExample.EXE
3.16227766016838
```

HARSHA

"System.Diagnostics.Process" class

System.Diagnostics.Process

- "Process" is a pre-defined class in "System.Diagnostics" namespace.
- This class's object represents an exe file (executable file).
- "Process" class is used to run an exe file.
- We can pass parameters to the exe file.

Syntax: System.Diagnostics.Process.Start("exe file path", parameters);

"System.Diagnostics.Process" class - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Console Application".
- Type the project name as "ProcessExample".
- Type the location as "C:\CSharp".
- Type the solution name as "ProcessExample".
- Click on OK.

Program.cs

```
using System;
using System.Diagnostics;
namespace ProcessExample
{
```

```

class Program
{
    static void Main()
    {
        //call "System.Diagnostics.Process.Start" method to run calculator
        Process.Start(@"C:\windows\system32\calc.exe");

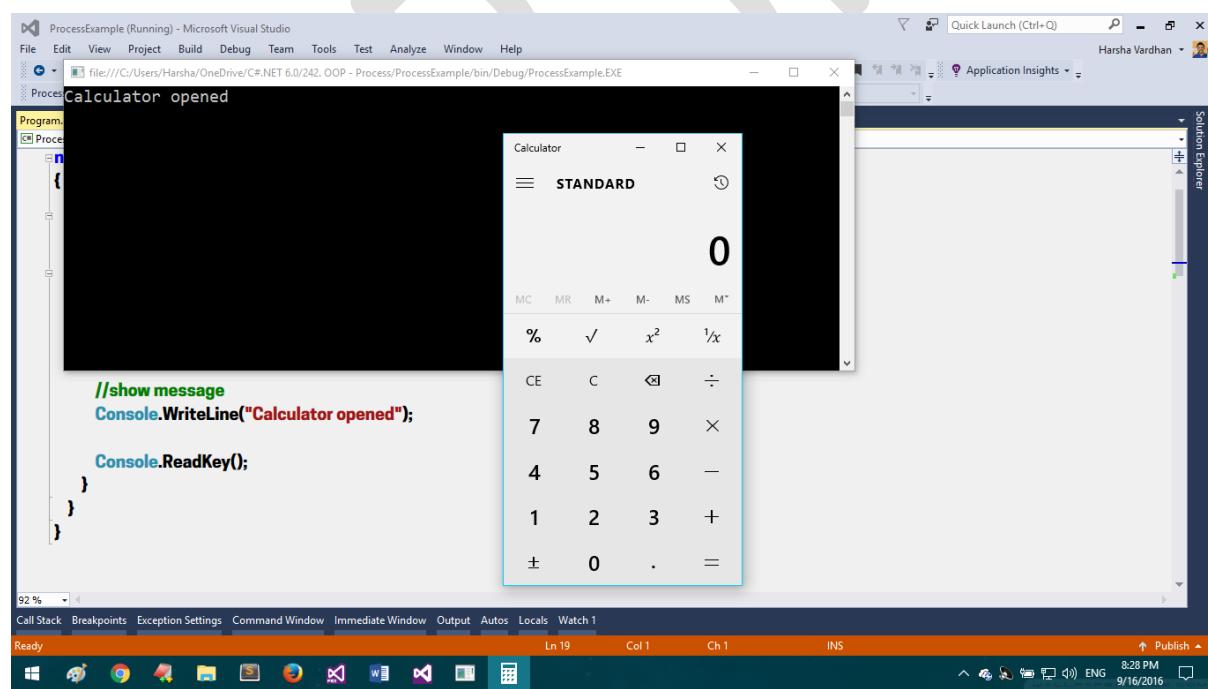
        //show message
        Console.WriteLine("Calculator opened");
        Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Command Line Arguments

Command Line Arguments

- It is used to receive one or more argument values from command prompt (command line).
- All the command line arguments will be received by the Main method as a string array.
- This technique is used in most of the windows applications such as Notepad, MS Office Word, VLC Media Player etc.
- Syntax: filename.exe value1 value2 ...

Command Line Arguments - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “App1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “App1”.
- Click on OK.

Program.cs

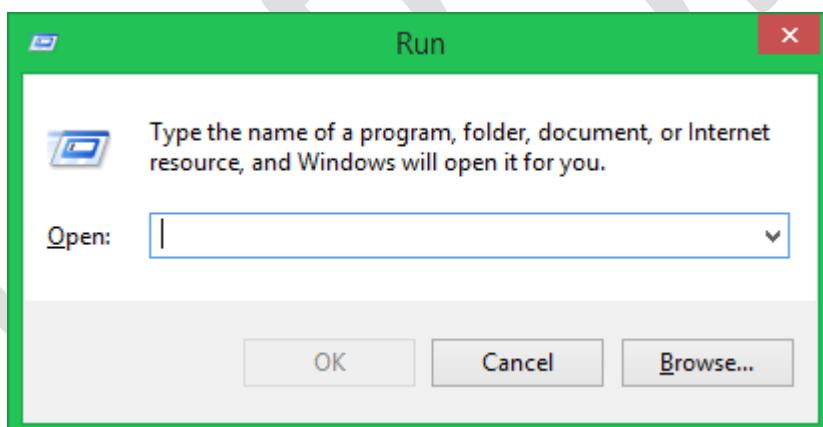
```
using System;
```

```
namespace App1
{
    class Program
```

```
{  
    static void Main(string[] args)  
    {  
        if (args.Length >= 2)  
        {  
            Console.WriteLine(args[0]);  
            Console.WriteLine(args[1]);  
        }  
        Console.ReadKey();  
    }  
}
```

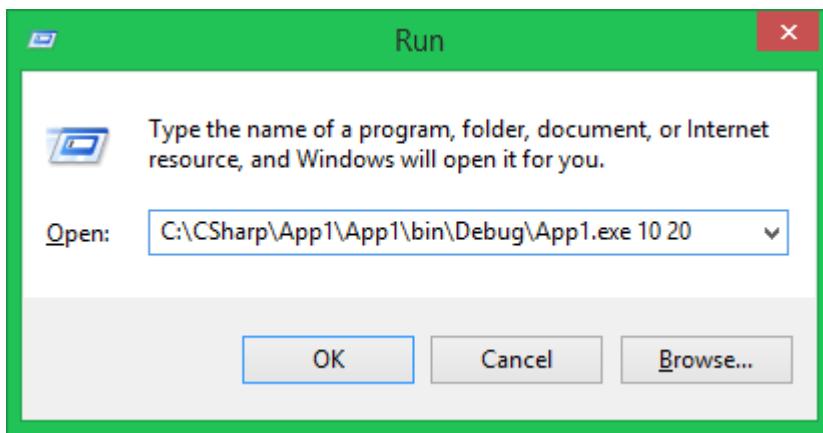
Running the project

- Press Windows + R to open “Run” window.



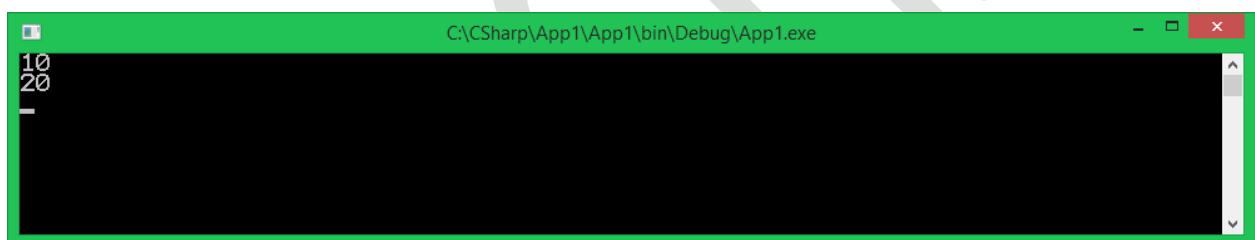
Type the following command.

C:\CSharp\App1\App1\bin\Debug\App1.exe 10 20



- Click on OK.

Output



Nullable Data Types

Nullable Data Types

- Null means “nothing” (or) “blank”.
- Data types are two types:
 - 1) Value-type data types (or) Structures (or) Non-nullable data types
 - The “structure variable” will be stored in “Stack” in RAM.
 - Null values are not supported.
 - 2) Reference-type data types (or) Classes (or) Nullable data types
 - The “objects” will be stored in “Heap” in RAM.
 - Null values are supported.

Syntax to convert non-nullable data type to nullable data type:

- *Structurename?* *variable* = null;
- [or]
- *Nullable<Structurename>* *variable* = null;

Nullable Data Types - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “NullableDataTypesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NullableDataTypesExample”.
- Click on OK.

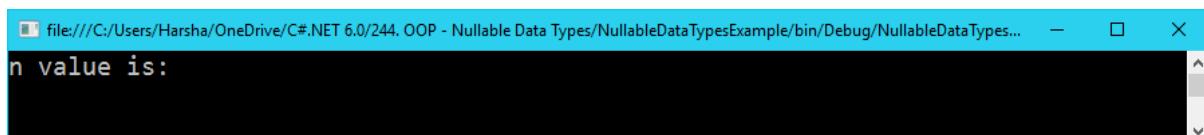
Program.cs

```
using System;
namespace NullableDataTypesExample
{
    class Program
    {
        static void Main()
        {
            //create a nullable variable
            int? n;
            //set "null" value into nullable variable
            n = null;
            //get the value of nullable variable
            Console.WriteLine("n value is: " + n);
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Exception Handling

Exception Handling

- Exception means “run time error”.
- When a statement can't be executed due to some reason, system treat it as a run time error (exception).
- When a runtime error found, automatically CLR raises the exception.
- Exceptions raise based on different factors like programmer's mistake, system problem, database connection or SQL statement execution problem etc.
- When exception is raised, the programs gets terminated un-expectedly (or) abruptly. It is inconvenience to the user. So the programmer has to use “exception handling” concept to avoid un-expected closing of the application.
- Exception handling is used to catch the run time errors (exceptions) and display the error message on the same page, instead of closing the application un-expectedly.

Syntax:

```
try
{
    Code here
}
catch (Exception ex)
{
    Code here
}
finally
{
    Code here
}
```

- “Try” block: Executes normally.
- “Catch” block: Executes only when the exception is raised in try block. It receives the exception details as “ex” object.
 - Message: Exception message
 - Stack Trace: Method called sequence
- “Finally” block”: Executes after completion of try block or catch block. Finally block is optional.

Exception Handling - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ExceptionHandlingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExceptionHandlingExample”.
- Click on OK.

Program.cs

```
using System;
```

```
namespace ExceptionHandlingExample
```

```
{  
    class Program  
    {  
        static void Main()  
        {  
            try  
            {  
                Console.Write("Enter first number: ");  
                int a = Convert.ToInt32(Console.ReadLine());  
  
                Console.Write("Enter second number: ");  
                int b = Convert.ToInt32(Console.ReadLine());  
  
                int c = a / b;  
  
                Console.WriteLine("Quotient is " + c);  
            }  
            catch (Exception ex)  
            {  
                Console.WriteLine(ex.Message);  
            }  
            finally  
            {  
                Console.WriteLine("Finished");  
            }  
            Console.ReadKey();  
        }  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/245. OOP - Exception Handling/ExceptionHandlingExample/bin/Debug/ExceptionHandlin... - X
Enter first number: 10
Enter second number: gfe34rt45y
Input string was not in a correct format.
Finished
```

Destructor

- Destructor is a special method of the class, which will be automatically called before deleting each object of the class.
- By using “destructors”, C#.NET allows the programmer to execute any code, just before deleting an object.
- Destructor’s name should be same as current class name.
- Destructor can’t have arguments and return value.
- A class can have only one destructor.
- Destructor can’t be overloaded.

Syntax of destructor:

```
~classname()
{
}
```

Destructor - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DestructorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DestructorExample”.
- Click on OK.

Program.cs

```
using System;

namespace DestructorExample
{
    class Class1
    {
        //constructor
        public Class1()
        {
            Console.WriteLine("Constructor");
            //Ex: opening the db connection
        }

        //destructor
        ~Class1()
        {
            Console.WriteLine("Destructor");
            //Ex: closing the db connection
        }
    }

    class Program
    {
        static void Main()
        {
            Class1 c1 = new Class1(); //constructor will be called here
            //lot of code
            //db work is over

            Console.ReadKey();
        } //destructor will be called here
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start without Debugging”.

Output



```
cmd C:\WINDOWS\system32\cmd.exe
Constructor
Destructor
Press any key to continue . . .
```

The "System.IDisposable" interface

System.IDisposable

- “System.IDisposable” is a pre-defined interface, which contains only one abstract method called “Dispose”. The “Dispose” method will be called automatically, when “using” statement is used while creating object. In the “Dispose” method, you can perform any cleaning process such as closing database connections or closing files etc.
- The benefit of using “Dispose” method instead of “destructor” is: the “Dispose” method will be called automatically as soon as “using” statement ends; then the cleaning process will be performed through “Dispose” method; so that we release the resources such as database connections and file connections; no need to wait until the end of the application.

Syntax of inhering from “System.IDisposable” interface:

```
class classname: System.IDisposable
{
    public void Dispose()
    {
        referencevariable = null;
    }
}
```

Syntax of inhering from “System.IDisposable” interface:

```
using (classname referencevariable = new classname())
{
}
```

The "System.IDisposable" interface - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IDisposableExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IDisposableExample”. Click on OK.

Program.cs

```
using System;
namespace IDisposableExample
{
    class Class1 : System.IDisposable
    {
        public Class1()
        {
            Console.WriteLine("object created");
            //db connection opened
        }
        public void method1()
        {
            Console.WriteLine("method1");
        }
        public void Dispose()
        {
            Console.WriteLine("disposed");
            //db connection closed
        }
        ~Class1()
        {
            //destructor
        }
    }
}
```

```
Console.WriteLine("destructor");
}
}
class Program
{
    static void Main()
    {
        using (Class1 c1 = new Class1())
        {
            c1.method1();
        } //Dispose

        Console.ReadKey();
    } //Destructor
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start without Debugging”.

Output



```
C:\WINDOWS\system32\cmd.exe
object created
method1
disposed
destructor
Press any key to continue . . .
```

Garbage Collection

- Garbage collection is a process of deleting unreferenced variables or objects from memory.
- Garbage collection will be performed automatically; but it will be performed at the end of main method only.
- To perform garbage collection immediately, use `System.GC.Collect()` method.
- Syntax: `System.GC.Collect();`

Garbage Collection - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “GarbageCollectionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “GarbageCollectionExample”.
- Click on OK.

Program.cs

```
using System;
```

```
namespace GarbageCollectionExample
{
    class Class1
    {
```

```
~Class1()
{
    Console.WriteLine("Destructor");
}
}

class Program
{
    static void Main()
    {
        //create two objects
        Class1 c1 = new Class1();
        Class1 c2 = new Class1();

        c1 = null;

        //delete all the objects
        GC.Collect();

        Console.WriteLine("Garbage collection done.");

        Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start without Debugging”.

Output



```
C:\WINDOWS\system32\cmd.exe
Garbage collection done.
Destructor
Destructor
Press any key to continue . . .
```

Delegates

Delegates

- Delegate is a reference variable that contains reference of one or more methods that have same signature.
 - Single Cast Delegates: Contains reference of only one method.
 - Multi Cast Delegates: Contains reference of multiple methods.
- We can call the target method using delegate.
- We don't use delegates individually; we use delegates only in events.
- The "delegate type" is used to specify signature of the target method.

Steps for development of delegates:

- Create a target method:

```
accessmodifier returntype methodname(arguments)
{
}
```
- Create delegate type (in the namespace):
`public delegate returntype delegatename(arguments);`
- Create a delegate (in the class or method);
`delegatename delegatename;`
- Store the address of target method in the delegate;
`delegatename = methodname;`
- Call the method indirectly using delegate;
`delegatename(value1, value2, ..);`

Single Cast Delegates - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SingleCastDelegatesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SingleCastDelegatesExample”. Click on OK.

Program.cs

```
using System;

namespace SingleCastDelegatesExample
{
    class Class1
    {
        //create target method
        public void Add(int a, int b)
        {
            int c = a + b;
            Console.WriteLine("Sum is " + c);
        }
    }

    //create a delegate type
    public delegate void MyDelegateType(int a, int b);

    class Program
    {
```

```
static void Main()
{
    //create a delegate
    MyDelegateType d;

    //create object
    Class1 c1 = new Class1();

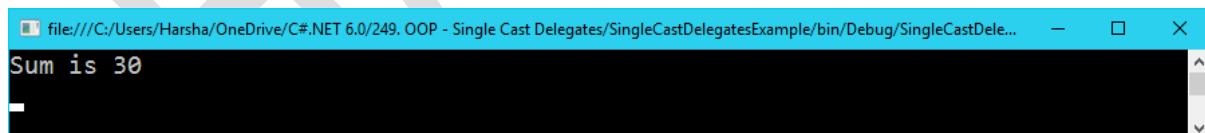
    //store the address of Add method into delegate
    d = c1.Add;

    //call the Add method using delegate
    d(10, 20);
    Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start without Debugging”.

Output



```
Sum is 30
```

Multi Cast Delegates - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MultiCastDelegatesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultiCastDelegatesExample”.
- Click on OK.

Program.cs

```
using System;

namespace MultiCastDelegatesExample
{
    class Class1
    {
        //create target method 1
        public void Add(int a, int b)
        {
            int c = a + b;
            Console.WriteLine("Sum is " + c);
        }

        //create target method 2
        public void Subtract(int a, int b)
        {
            int c = a - b;
            Console.WriteLine("Difference is " + c);
        }
}
```

```

        }
    }

//create a delegate type
public delegate void MyDelegateType(int a, int b);

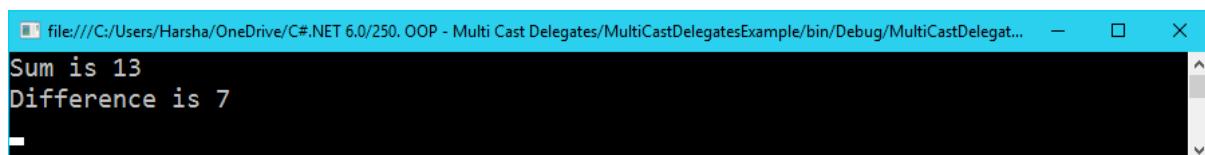
class Program
{
    static void Main()
    {
        //create a delegate
        MyDelegateType d;
        //create object
        Class1 c1 = new Class1();
        //store the address of Add method into delegate
        d = c1.Add;
        //store the address of Subtract method also into the delegate
        d += c1.Subtract;
        //call the Add and Subtract methods using delegate
        d(10, 3);
        Console.ReadKey();
    }
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start without Debugging”.

Output



```
Sum is 13
Difference is 7
```

Events

- Events are used to implement “publisher – subscriber” model.
- “Publisher” is a class that contains an event and also raises the event.
- “Subscriber” is a class that subscribes to the event of publisher class. That means subscriber class provides an event handler that can be attached to the event.
- When the event is raised in the publisher class, the event handler method will be executed.
- An event can be subscribed with multiple methods also.
- In real time, publisher class will be developed by one programmer and subscriber class will be developed by other programmer.

Steps for development of events:

- Create a delegate type (in the namespace):

```
public delegate returntype delegatename(argument1, argument2, ...)
```

- Create publisher class:

```
class publisherclassname
{
}
```

- Create an event in the publisher class:

```
public event delegatename eventname;
```

- Raise the event in the publisher class:

```
eventname (value1, value2, ...)
```

- Create subscriber class:

```
class subscriberclassname
{
}
```

- Create the event handler method in subscriber class:

```
public returnType methodname (arguments)
{
}
```

- Attach the event handler method to the event:

```
eventname += methodname ;
```

Events - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “EventsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EventsExample”. Click on OK.

Program.cs

```
using System;

namespace EventsExample
{
    public delegate void mydelegatename(string username);

    class publisherclass
    {
        public event mydelegatename myevent;
```

```
public void raiseevent(string username)
{
    myevent(username); //raise event
}
}

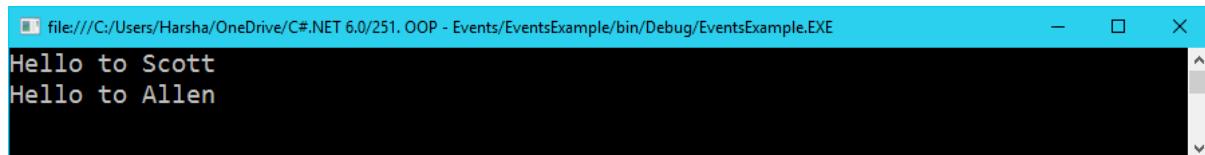
class subscriberclass
{
    public void subscribedmethod(string username)
    {
        Console.WriteLine("Hello to " + username);
    }
    public void connect()
    {
        publisherclass pc = new publisherclass();
        pc.myevent += subscribedmethod;

        pc.raiseevent("Scott");
        pc.raiseevent("Allen");
    }
}
class Program
{
    static void Main()
    {
        subscriberclass sc = new subscriberclass();
        sc.connect();
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a terminal window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/251. OOP - Events/EventsExample/bin/Debug/EventsExample.EXE". The window contains the text "Hello to Scott" and "Hello to Allen", demonstrating the execution of a C# program.

HARSHA

Anonymous Methods

- Anonymous methods are used to quickly create a method inside another method.
- Anonymous methods are useful while handling the events.
- Anonymous methods can't be a member of the class.
- Anonymous methods will have no name.
- Anonymous method's address should be stored in a delegate (or) event, through the delegate or event only we can call it.

Syntax of anonymous method:

```
delegate / event = delegate(argument1, argument2, ...)
{
    Code here
}
```

Anonymous Methods - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “AnonymousMethodsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AnonymousMethodsExample”.
- Click on OK.

Program.cs

```
using System;
namespace AnonymousMethodsExample
{
    class publisherclass
    {
        public event mydelegatetype myevent;

        public int x = 0, y = 0;

        public void increment()
        {
            x += 5;
            y += 5;
            myevent(x, y);
        }
    }

    class subscriberclass
    {
        public void dowork()
        {
            publisherclass pc = new publisherclass();
            pc.myevent += delegate(int a, int b)
            {
                int c = a + b;
                Console.WriteLine("Sum is " + c);
            };
            pc.increment();
        }
    }
}

delegate void mydelegatetype(int a, int b);

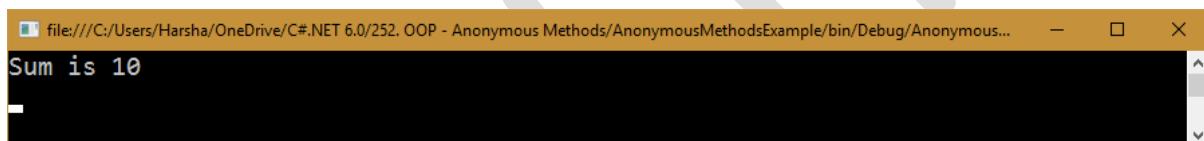
class Program
```

```
{  
    static void Main()  
    {  
        subscriberclass sc = new subscriberclass();  
        sc.dowork();  
        Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Lambda Expressions

- Lambda Expressions are used to quickly create a method inside another method.
- Lambda Expressions are useful while handling the events.
- Lambda Expressions can't be a member of the class.
- Lambda Expressions will have no name.
- Lambda Expressions's address should be stored in a delegate (or) event, through the delegate or event only we can call it.

Syntax of Lambda Expressions:

```
delegate / event = (argument1, argument2, ...) =>
{
    Code here
}
```

Lambda Expressions - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “LambdaExpressionsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LambdaExpressionsExample”.
- Click on OK.

Program.cs

```
using System;
namespace LambdaExpressionsExample
{
    class publisherclass
    {
        public event mydelegatetype myevent;
        public int x = 0, y = 0;
        public void increment()
        {
            x += 5;
            y += 5;
            myevent(x, y);
        }
    }
    class subscriberclass
    {
        public void dowork()
        {
            publisherclass pc = new publisherclass();
            pc.myevent += (a, b) =>
            {
                int c = a + b;
                Console.WriteLine("Sum is " + c);
            };
            pc.increment();
        }
    }
}

delegate void mydelegatetype(int a, int b);
class Program
{
    static void Main()
    {
        subscriberclass sc = new subscriberclass();
        sc.dowork();
        Console.ReadKey();
    }
}
```

```
}
```

```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/253. OOP - Lambda Expressions/LambdaExpressionsExample/bin/Debug/LambdaExpressio... - □ ×  
Sum is 10  
-
```

Inline Lambda Expressions

- Inline Lambda Expressions are similar to lambda expressions, but can be used only for the methods that receives one or more arguments and return a value.
- Inline lambda expressions can contain only just a single calculation; can't contain any other additional code.
- Inline lambda expressions can receive one or more arguments; must return a value.

Syntax of Inline Lambda Expressions:

```
delegate / event - (argument1, argument2, ...) =>
(
    Calculation here
)
```

Inline Lambda Expressions - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InlineLambdaExpressionsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InlineLambdaExpressionsExample”.
- Click on OK.

Program.cs

```
using System;
namespace InlineLambdaExample
{
    class publisherclass
    {
        public event mydelegatetype myevent;

        public int x = 0, y = 0;

        public void increment()
        {
            x += 5;
            y += 5;
            int result = myevent(x, y);
            Console.WriteLine("Sum is " + result);
        }
    }
    class subscriberclass
    {
        public void dowork()
        {
            publisherclass pc = new publisherclass();
            pc.myevent += (a, b) => (a + b);
            pc.increment();
        }
    }
    delegate int mydelegatetype(int a, int b);

    class Program
    {
        static void Main()
        {
            subscriberclass sc = new subscriberclass();
            sc.dowork();
            Console.ReadKey();
        }
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Inner Classes

- Class inside another class is called as “inner class”.
- The class that encloses inner class, is called as “outer class”.
- Use “inner class”, if you want to specify that the class is a part of the outer class logically.

Syntax to create inner class

```
class outerclassname  
{  
    public class innerclassname  
    {  
    }  
}
```

Syntax to access the inner class

```
outerclassname . innerclassname
```

Inner Classes - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InnerClassesExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “InnerClassesExample”.
- Click on OK.

Program.cs

```
using System;

namespace InnerClassesExample
{
    class Class1
    {
        public class Class2
        {
            public void mymethod()
            {
                Console.WriteLine("mymethod");
            }
        }
    }

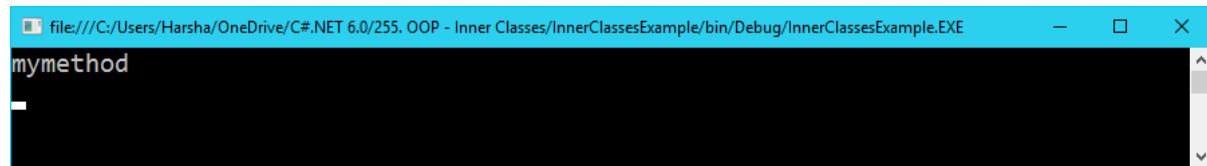
    class Program
    {
        static void Main()
        {
            Class1.Class2 c2 = new Class1.Class2();
            c2.mymethod();

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
mymethod
```

HARSHA

Indexer

Indexer

- Indexer makes it easy to access a member array through the reference variable of a class.
- Indexer is a property with name “this”.
- Indexer overloading is possible. That means we can create multiple indexers in the same class, with different arguments.
- Mostly indexers are used in pre-defined classes; it's rare to use indexers in user-defined classes in real time.

Syntax to create indexer:

```
class classname
{
    public datatype this[string variablename]
    {
        get
        {
            return value;
        }
        set
        {
            Field = value;
        }
    }
}
```

Syntax to access the indexer:

Referencevariable["string value"]

Indexer - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “IndexerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “IndexerExample”.
- Click on OK.

Program.cs

```
using System;

namespace IndexerExample
{
    class Class1
    {
        //creating an array
        public string[] cities = new string[] { "Hyderabad", "Chennai",
        "Bangalore" };
        public long[] population = new long[] { 9000, 5000, 10000 };

        //indexer
        public long this[string city]
        {
            get
            {
                int n = Array.IndexOf(cities, city);
                return population[n];
            }
        }
    }
}
```

```
        }
    }
}

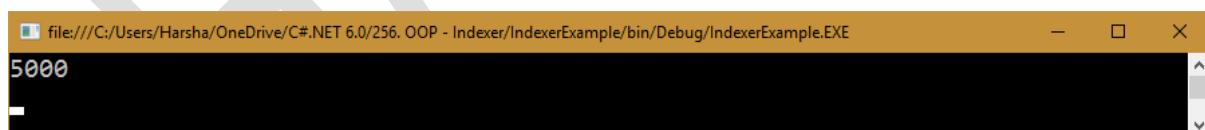
class Program
{
    static void Main()
    {
        Class1 c1 = new Class1();
        long p = c1["Chennai"];
        Console.WriteLine(p);

        //Console.Write(c1.population[Array.IndexOf(c1.cities, "Chennai")]);
        Console.ReadKey();
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Assemblies

- In .NET, an “Assembly” is a file that contains “compiled source code (in MSIL language) of the .net program”.
- Assemblies can’t be run directly based on the operating system; can only be run by CLR (Common Language Runtime). CLR reads MSIL code from assembly and convert the same into native machine language automatically, while .net program execution.
- Assemblies are two types:
 1. Executable assemblies
 2. Library assemblies

Executable Assemblies

- “Executable assemblies” are “EXE files” that can run directly (based on CLR).
- When you compile the following types of applications, Visual Studio automatically generates (creates) an “Executable Assembly”:
 - Console Application
 - Windows Forms Application
 - Windows Service
- Drawback: The code of executable assemblies (projects) can’t be called in another assemblies (projects).

Library Assemblies

- “Library Assemblies” are “DLL files” that can’t run directly. The library assemblies are always “supporting files” to “Executable Assembly”. Executable assemblies dynamically call the library assemblies at run time.

- When you compile the following types of applications, Visual Studio automatically generates (creates) an “Library Assembly”:
 - Class Library
 - Windows Forms Control Library
- An “executable assembly” can call any no. of library assemblies.
- A “library assembly” can be called by any no. of executable assemblies.

Types of Library Assemblies

- Library assemblies are divided into two types, based on where the library assembly is placed.
- Library assemblies are two types:
 1. Private Assemblies
 2. Shared Assemblies

Private Assemblies

- Private assemblies (dll files) should be placed in each “application folder”.
- Private assemblies are faster in execution but there is memory wastage, if multiple applications are consuming the same private assembly and are running on same system.

Shared Assemblies

- Shared assemblies are located in GAC (Global Assembly Cache) folder and can be accessed by the applications that are running on the same system.
- Shared assemblies save memory, because there is no need of duplicate copies of dll file in every app’s folder.

- Shared assemblies are a bit slower than private assemblies.

GAC

- GAC (Global Assembly Cache) is a special folder in .NET Framework, which contains all the shared assemblies (shared dll files).
- The shared dll files that are present at GAC folder can be accessible from any application running in the same system.
- GAC Folder: C:\Windows\Microsoft.NET\assembly\GAC_MSIL
- Advantage of GAC folder: Avoid the “DLL hell”.
- DLL hell: When we install a new version of an application, the dll files of old version of the application will be overwritten with latest version’s dll files. GAC folder solves this problem with unique identification of dll files.

SNK

- An assembly must have “snk” (Strong Name Key) to be placed in GAC folder.
- SNK (Strong Name Key) file is contains “unique code”, which is useful for “unique identification” of dll files that are placed in GAC folder.
- SNK file can be created in visual studio through the “project properties” – “Signing” option.

Class Library

- “Class Library” is a special type of project in Visual Studio.
- Class Library is a collection of re-usable classes.
- When the class library project is compiled, a “library assembly” (DLL file) will be generated.

Class Libraries & Private Assemblies - Example

Creating Class Library

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Class Library”.
- Type the project name as “ClassLibrary1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ClassLibrary1”.
- Click on OK.

Class1.cs

```
using System;

namespace ClassLibrary1
{
    public class Class1
    {
        public bool CheckEmail(string Email)
        {
            bool b;
            if (Email.Contains(" ") == false && Email.IndexOf("@") > 0)
            {
                b = true;
            }
            else
            {
                b = false;
            }
            return b;
        }
    }
}
```

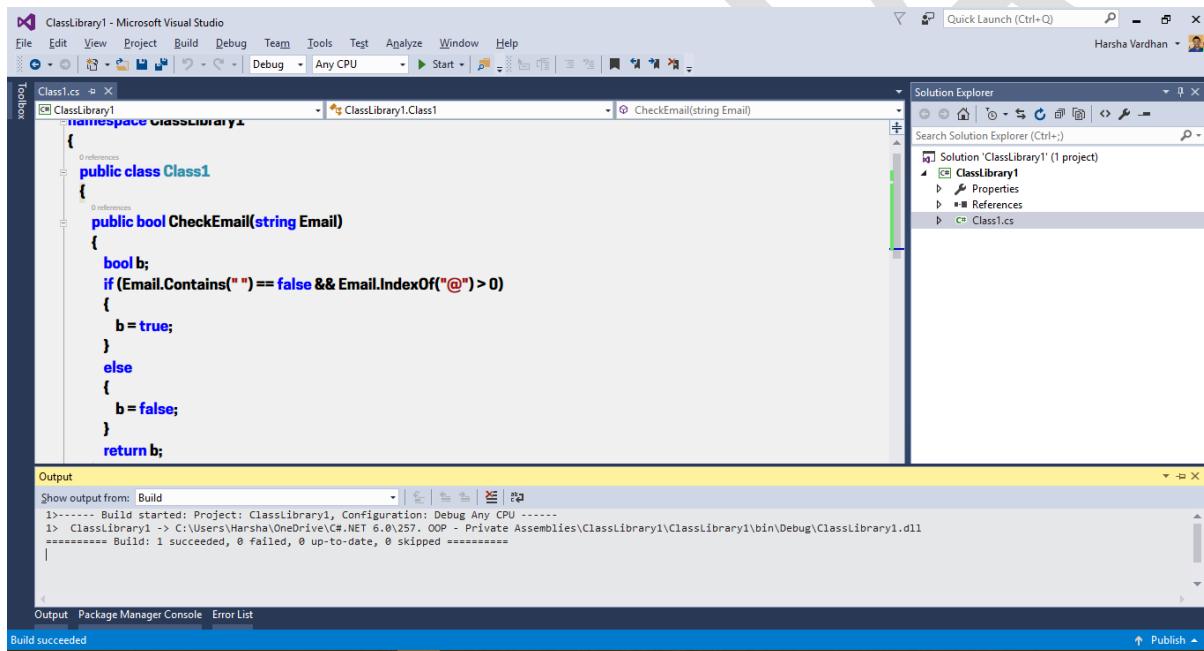
```
}
```

```
}
```

```
}
```

Compiling the project

- Go to “Build” menu and click on “Build Solution”.
- Note: You can't run the class library directly.
- Path of dll file: C:\CSharp\ClassLibrary1\ClassLibrary1\bin\Debug\ClassLibrary1.dll



Creating the client project (Console Application):

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ConsoleApplication1”.

- Type the location as “C:\CSharp”.
- Type the solution name as “ConsoleApplication1”.
- Click on OK.

Add reference

- Open solution explorer.
- Right click on the project name (ConsoleApplication1) and click on “Add” – “Reference”.
- Click on “Browse”.
- Select the dll file (C:\CSharp\ClassLibrary1\ClassLibrary1\bin\Debug\ClassLibrary1.dll).
- Click on “Add” – “OK”.
- Then it shows the “ClassLibrary1” in the “References” list in the solution explorer.

Program.cs

```
using System;
using ClassLibrary1;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main()
        {
            Class1 c1 = new Class1();
            string s = "sample@gmail.com";
            bool result = c1.CheckEmail(s);
            if (result == true)
            {
                Console.WriteLine(s);
                Console.WriteLine("Valid email address");
            }
        }
    }
}
```

```
else
{
    Console.WriteLine(s);
    Console.WriteLine("Invalid email address");
}
Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Shared Assemblies - Example

Creating Class Library

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Class Library”.
- Type the project name as “ClassLibrary2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ClassLibrary2”.
- Click on OK.

Class1.cs

```
using System;

namespace ClassLibrary2
{
    public class Class1
    {
        public bool CheckEmail(string Email)
        {
            bool b;
            if (Email.Contains(" ") == false && Email.IndexOf("@") > 0)
            {
                b = true;
            }
            else
            {
                b = false;
            }
            return b;
        }
    }
}
```

```
}
```

```
}
```

```
}
```

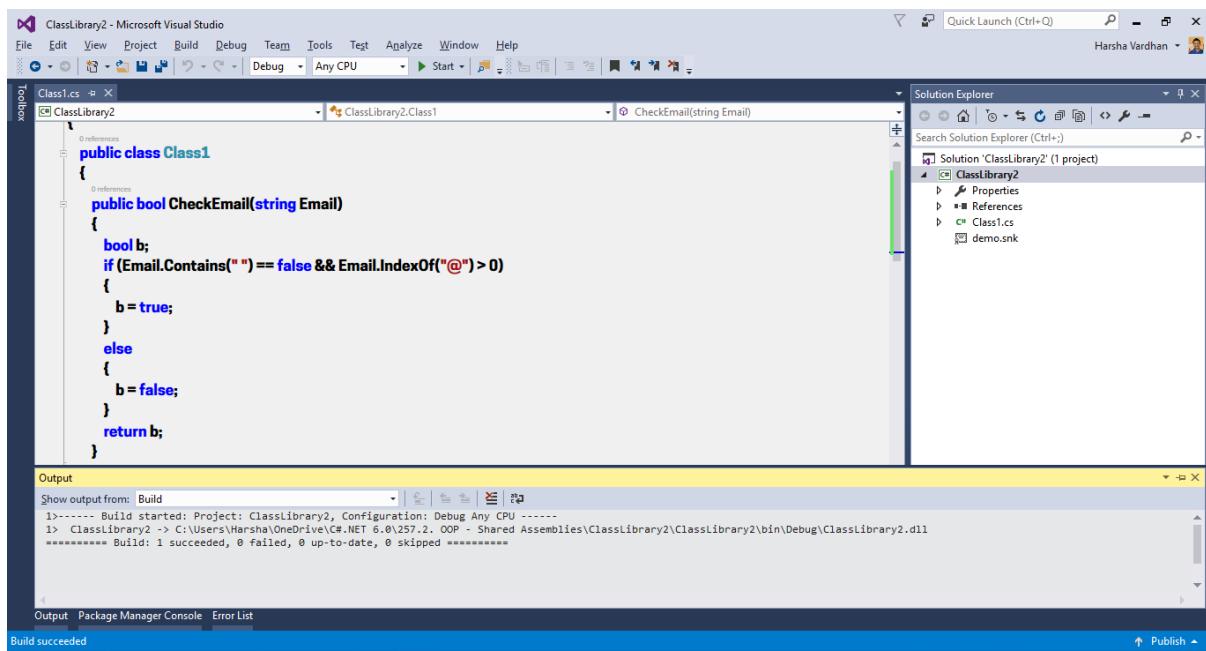
Creating “snk” file:

- To create snk file, open solution explorer; right click on the project name “ClassLibrary2” and click on “Properties”.
- Click on “Signing”. Check the checkbox “Sign the assembly”.
- Click on the dropdown list and click on “New”.
- Type the file name as “demo”. Uncheck the checkbox “protect my key file with a password”. Click on OK.
- Close the properties window.

Compiling the project

- Go to “Build” menu and click on “Build Solution”.
- Note: You can't run the class library directly.

Path of dll file: C:\CSharp\ClassLibrary2\ClassLibrary2\bin\Debug\ClassLibrary2.dll



Uploading the dll file into GAC

- Go to “Start” – “All Programs” – “Visual Studio 2015” – Right click on “Developer Command Prompt for VS 2015” and click on “Run as administrator”. Click on “Yes”.
- Type the following command in the console window:

```
gacutil -i "C:\CSharp\ClassLibrary2\ClassLibrary2\bin\Debug\ClassLibrary2.dll"
```

- Press Enter.
- It shows the following message:
Assembly successfully added to the cache.

Creating the client project (Console Application):

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ConsoleApplication2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ConsoleApplication2”.
- Click on OK.

Add reference

- Open solution explorer.
- Right click on the project name (ConsoleApplication2) and click on “Add” – “Reference”.
- Click on “Browse”.
- Select the dll file (C:\CSharp\ClassLibrary2\ClassLibrary2\bin\Debug\ClassLibrary2.dll).
- Click on “Add” – “OK”.
- Then it shows the “ClassLibrary2” in the “References” list in the solution explorer.
- Note: You can notice that the “ClassLibrary2.dll” file is not present in “C:\CSharp\ConsoleApplication2\ConsoleApplication2\bin\Debug” folder. Even it is present, the “ConsoleApplication2.exe” file works without “ClassLibrary2.dll” file present in the same folder, because it will be accessed from the GAC folder.

Program.cs

```
using System;
using ClassLibrary2;

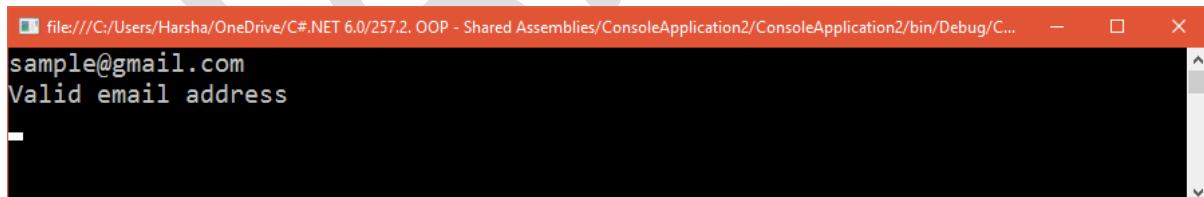
namespace ConsoleApplication2
{
    class Program
    {
        static void Main()
        {
            Class1 c1 = new Class1();
```

```
string s = "sample@gmail.com";
bool result = c1.CheckEmail(s);
if (result == true)
{
    Console.WriteLine(s);
    Console.WriteLine("Valid email address");
}
else
{
    Console.WriteLine(s);
    Console.WriteLine("Invalid email address");
}
Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/257.2. OOP - Shared Assemblies/ConsoleApplication2/ConsoleApplication2/bin/Debug/C...
sample@gmail.com
Valid email address
```

Documentation Comments

- Documentation comments are used to provide description for the methods and its arguments, which will appear while calling the method (while typing the code in visual studio).

Syntax for method description:

```
/// <summary>Method description here</summary>
```

Syntax for parameter description:

```
/// <param name="parameter name">Parameter description here</param>
```

Documentation Comments - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DocumentationCommentsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DocumentationCommentsExample”.
- Click on OK.

Program.cs

```
using System;

namespace DocumentationCommentsExample
{
    class Class1
    {
        /// <summary>Adds two numbers</summary>
        /// <param name="firstNumber">First number for adding</param>
        /// <param name="secondNumber">Second number for
        adding</param>
        public void Add(int firstNumber, int secondNumber)
        {
            //code here
        }
    }

    class Program
    {
        static void Main()
        {
            Class1 c1 = new Class1();

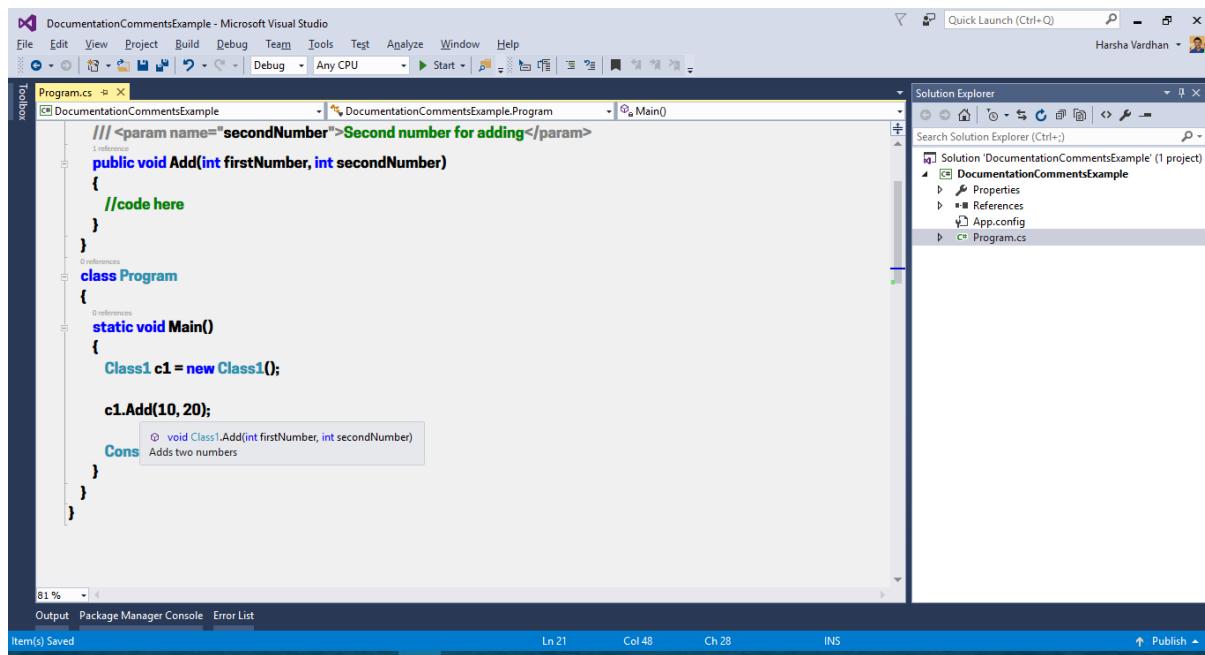
            c1.Add(10, 20);

            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows the Microsoft Visual Studio interface with the following details:

- Solution Explorer:** Shows a single project named "DocumentationCommentsExample" with files like "Properties", "References", "App.config", and "Program.cs".
- Code Editor (Program.cs):** Displays C# code. A tooltip is visible over the line `Cons` in the `Main` method, containing the text "Adds two numbers".
- Status Bar:** Shows "Item(s) Saved", "Ln 21", "Col 48", "Ch 28", and "INS".

While typing the code in visual studio, it shows the description as tooltip.

Extension Methods

- Extension methods are methods that are indirectly injected into a compiled class, without modifying its source code.
- Assume, there is a re-usable class in the class library and its reference is added in the client project.
- From the class library, you can add a new method to the re-usable class, without modifying the source code of the re-usable class.
- That method is called as “extension method”.
- Extension method should be a static method; it should be inside a static class.
- It should be public method.
- It should have at least one argument of the target class type with “this” keyword. That argument represents “this” keyword in the extension method. “This” keyword represents the current object, based on which the method is called.
- The extension method can have maximum any no. of arguments, but minimum is one.
- The extension method can have any return type.

Syntax of extension method (in the class library):

```
public static class classname
{
    public static returntype methodname(this classname
argumentname)
    {
        Code here
    }
}
```

Syntax of call the extension method (in the client app):

referencevariablename.methodname (arguments);

Extension Methods - Example

Creating Class Library

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Class Library”.
- Type the project name as “ClassLibrary1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ClassLibrary1”.
- Click on OK.

Class1.cs

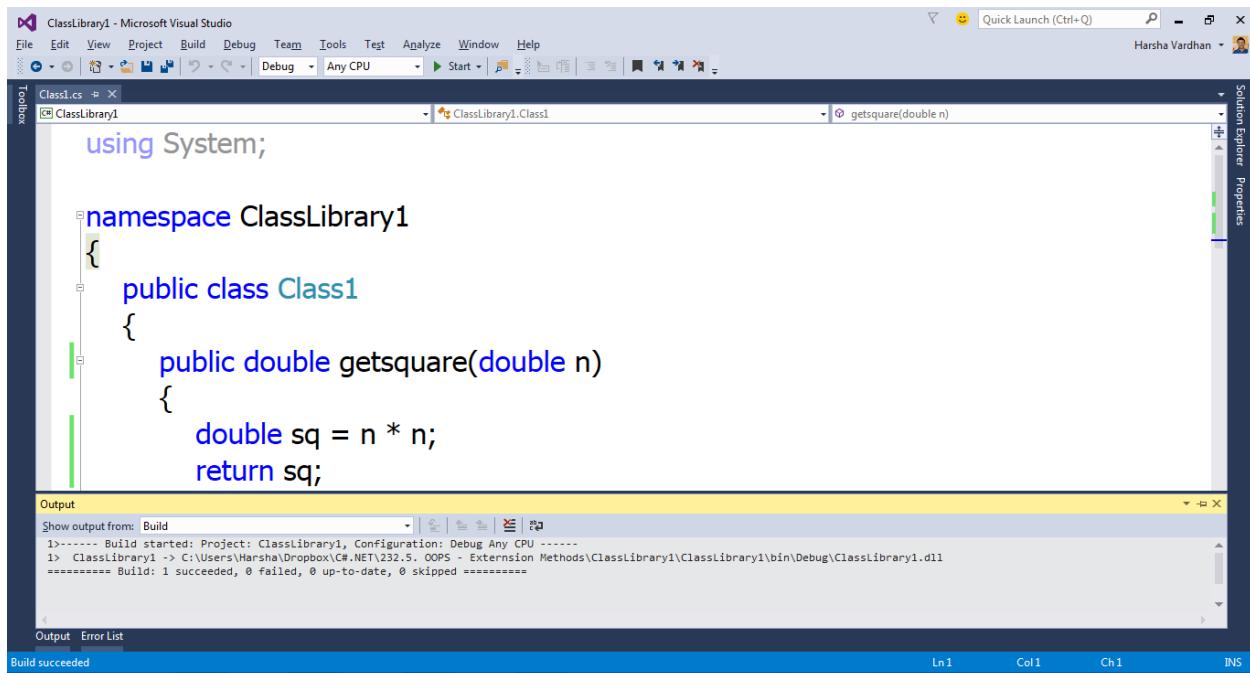
```
using System;

namespace ClassLibrary1
{
    public class Class1
    {
        public double getsquare(double n)
        {
            double sq = n * n;
            return sq;
        }
    }
}
```

Compiling the project

- Go to “Build” menu and click on “Build Solution”.

- Note: You can't run the class library directly.
- Path of dll file: C:\CSharp\ClassLibrary1\ClassLibrary1\bin\Debug\ClassLibrary1.dll



```
Class1.cs - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Toolbox
ClassLibrary1 - ClassLibrary1.cs
Solution Explorer Properties
Class1.cs
using System;
namespace ClassLibrary1
{
    public class Class1
    {
        public double getsquare(double n)
        {
            double sq = n * n;
            return sq;
        }
    }
}
Output
Show output from: Build
1>----- Build started: Project: ClassLibrary1, Configuration: Debug Any CPU -----
1> Classlibrary1 -> C:\Users\Harsha\Dropbox\C#\NET\232.5. OOPS - Extension Methods\ClassLibrary1\ClassLibrary1\bin\Debug\ClassLibrary1.dll
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
Output Error List
Build succeeded
Ln1 Col1 Ch1 INS
```

Creating the client project (Console Application):

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ConsoleApplication1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ConsoleApplication1”.
- Click on OK.

Add reference

- Open solution explorer.
- Right click on the project name (ConsoleApplication1) and click on “Add” – “Reference”.
- Click on “Browse”.
- Select the dll file (C:\CSharp\ClassLibrary1\ClassLibrary1\bin\Debug\ClassLibrary1.dll).
- Click on “Add” – “OK”.
- Then it shows the “ClassLibrary1” in the “References” list in the solution explorer.

Program.cs

```
using System;
using ClassLibrary1;

namespace ConsoleApplication1
{
    static class Class2
    {
        public static double getcube(this Class1 c1, double n)
        {
            double cb = n * n * n;
            return cb;
        }
    }

    class Program
    {
        static void Main()
        {
            Class1 c1 = new Class1();
            int a = 10;

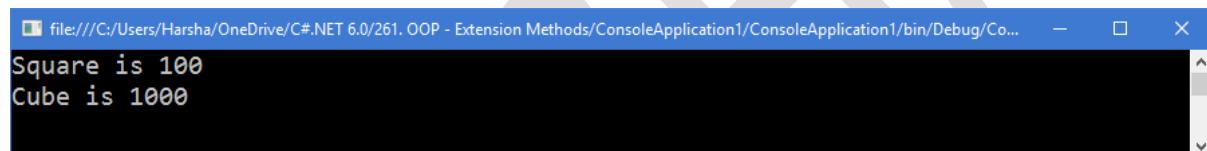
            double result1 = c1.getsquare(a);
            Console.WriteLine("Square is " + result1);
        }
    }
}
```

```
    double result2 = c1.getcube(a);
    Console.WriteLine("Cube is " + result2);
    Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/261. OOP - Extension Methods/ConsoleApplication1/ConsoleApplication1/bin/Debug/Co...
Square is 100
Cube is 1000
```

C# – Console - System.IO Namespace

Introduction to "System.IO" namespace

- The “System.IO” namespace provides a set of classes to manipulate files and folders in the hard disk.

List of important classes in System.IO namespace:

1. System.IO.FileInfo
2. System.IO.DirectoryInfo
3. System.IO.File
4. System.IO.Directory
5. System.IO.FileStream
6. System.IO.StreamWriter
7. System.IO.StreamReader

The “System.IO.FileInfo” class

System.IO.FileInfo class

- The “FileInfo” is a class in “System.IO” namespace.
- The “FileInfo” class’s object represents a file in the harddisk.
- The “FileInfo” class is used to manipulate a file in the harddisk.

Steps for development of “FileInfo” class:

- Import the namespace:
 - `using System.IO;`
- Create a reference variable:
 - `FileInfo referencevariable;`
- Create an object:
 - `referencevariable = new FileInfo("file path");`
- Set properties:
 - `referencevariable.property = value;`
- Call method:
 - `referencevariable.methodname();`

Properties of “FileInfo” class:

Sl. No	Property	Description
1	DirectoryName	Represents the current file's location's full path. <u>Syntax:</u> <code>referencevariable.DirectoryName</code>
2	Name	Represents only name of the file with extension (without path). <u>Syntax:</u> <code>referencevariable.Name</code>
3	Extension	Represents file extension of current file. <u>Syntax:</u> <code>referencevariable.Extension</code>
4	FullName	Represents full path of the current file. <u>Syntax:</u> <code>referencevariable.FullName</code>
5	Exists	Represents whether the current file exists or not. It returns a Boolean value. True: The file exists False: The file not exists <u>Syntax:</u> <code>referencevariable.Exists</code>
6	Length	Represents file size (no. of bytes). <u>Syntax:</u> <code>referencevariable.Length</code>
7	CreationTime	Represents the date and time of file creation. <u>Syntax:</u> <code>referencevariable.CreationTime</code>
8	LastAccessTime	Represents the date and time of file access. <u>Syntax:</u> <code>referencevariable.LastAccessTime</code>
9	LastWriteTime	Represents the date and time of file last modification. <u>Syntax:</u> <code>referencevariable.LastWriteTime</code>

10	Attributes	Represents the attributes of the current file. <u>Syntax:</u> <i>referencevariable.Attributes</i>
11	IsReadOnly	Represents whether the current file is readonly or not. <u>Syntax:</u> <i>referencevariable.IsReadOnly</i>

Constructors of “FileInfo” class:

Sl. No	Constructor	Description
1	FileInfo()	Initializes the file path. <u>Syntax:</u> <i>new FileInfo(string FilePath)</i>

Methods of “FileInfo” class:

Sl. No	Method	Description
1	CopyTo()	Copies the current file into the specified destination path. It is equal to “copy and paste”. <u>Syntax:</u> <i>referencevariable.CopyTo(string DestinationFilePath)</i>
2	Delete()	Deletes the current file. <u>Syntax:</u> <i>referencevariable.Delete()</i>
3	MoveTo()	Moves the current file into the specified destination path. It is equal to “cut and paste”. <u>Syntax:</u> <i>referencevariable.MoveTo(string DestinationFilePath)</i>

The “System.IO.FileInfo” class - Example

Creating Project

- Create the following folders and files:

- C:\CSharp
 - Sample.pdf

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “FileInfoExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FileInfoExample”.
- Click on OK.

Program.cs

//Note: Create "C:\CSharp\sample.txt"

```
using System;
using System.IO;

namespace namespace1
{
    class Program
    {
        static void Main()
        {
```

```

FileInfo f;
f = new FileInfo(@"c:\CSharp\sample.txt");
Console.WriteLine("Exists: " + f.Exists);

if (f.Exists)
{
    Console.WriteLine("Full name: " + f.FullName);
    Console.WriteLine("Name: " + f.Name);
    Console.WriteLine("Directory name: " + f.DirectoryName);
    Console.WriteLine("Extension: " + f.Extension);
    Console.WriteLine("Creation date and time: " + f.CreationTime);
    Console.WriteLine("Modification date and time: " +
f.LastWriteTime);
    Console.WriteLine("Access date and time: " + f.LastAccessTime);
    Console.WriteLine("Length: " + f.Length + " bytes");
}
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/264. Console - IO - FileInfo/FileInfoExample/bin/Debug/FileInfoExample.EXE
Exists: True
Full name: c:\CSharp\sample.txt
Name: sample.txt
Directory name: c:\CSharp
Extension: .txt
Creation date and time: 9/22/2016 7:20:05 PM
Modification date and time: 7/1/2016 11:47:24 AM
Access date and time: 9/22/2016 7:20:05 PM
Length: 23 bytes
-
```

It shows the file details as above.

The “System.IO.DirectoryInfo” class

- The “DirectoryInfo” is a class in “System.IO” namespace.
- The “DirectoryInfo” class’s object represents a folder (directory) in the harddisk.
- The “DirectoryInfo” class is used to manipulate a folder (directory) in the harddisk.

Steps for development of “DirectoryInfo” class:

- Import the namespace:
 - `using System.IO;`
- Create a reference variable:
 - `DirectoryInfo referencevariable;`
- Create an object:
 - `referencevariable = new DirectoryInfo("folder path");`
- Set properties:
 - `referencevariable.property = value;`
- Call method:
 - `referencevariable.methodname();`

Properties of “ DirectoryInfo ” class:

Sl. No	Property	Description
1	Parent.FullName	Represents the current folder's parent folder's full path. <u>Syntax:</u> <i>referencevariable</i> .Parent.FullName
2	Name	Represents only name of the folder (without parent folder's path). <u>Syntax:</u> <i>referencevariable</i> .Name
3	FullName	Represents full path of the current folder. <u>Syntax:</u> <i>referencevariable</i> .FullName
4	Root.FullName	Represents full path of the root folder of the current folder. <u>Syntax:</u> <i>referencevariable</i> .Root.FullName
5	Exists	Represents whether the current folder exists or not. It returns a Boolean value. True: The folder exists False: The folder not exists <u>Syntax:</u> <i>referencevariable</i> .Exists
6	CreationTime	Represents the date and time of folder creation. <u>Syntax:</u> <i>referencevariable</i> .CreationTime
7	LastAccessTime	Represents the date and time of folder access. <u>Syntax:</u> <i>referencevariable</i> .LastAccessTime
8	LastWriteTime	Represents the date and time of folder last modification. <u>Syntax:</u> <i>referencevariable</i> .LastWriteTime
9	Attributes	Represents the attributes of the current folder. <u>Syntax:</u> <i>referencevariable</i> .Attributes

Constructors of “ DirectoryInfo ” class:

Sl. No	Constructor	Description
1	DirectoryInfo()	<p>Initializes the folder path.</p> <p><u>Syntax:</u> new DirectoryInfo(string FolderPath)</p>

Methods of “ DirectoryInfo ” class:

Sl. No	Method	Description
1	Delete()	<p>Deletes the current folder, if it is empty.</p> <p>If the current folder is not empty, it throws an exception automatically.</p> <p><u>Syntax:</u> referencevariable.Delete()</p>
2	Delete(true)	<p>Deletes the current folder, including all of its files and subfolders.</p> <p><u>Syntax:</u> referencevariable.Delete(true)</p> <p><u>Warning:</u> This action is permanent. It is impossible to undo this action. Check the folder path before running the program.</p>
3	MoveTo()	<p>Moves the current folder into the specified destination path.</p> <p>It is equal to “cut and paste”.</p> <p><u>Note:</u> The destination folder path should be in the same drive.</p> <p>Ex: C: drive</p> <p><u>Syntax:</u> referencevariable.MoveTo(string DestinationFolderPath)</p>

4	CreateSubdirectory()	Creates a new folder with specified name. <u>Syntax:</u> <code>referencevariable.CreateSubdirectory(string newfoldername)</code>
5	GetFiles()	Returns the list of files of the current folder, as an array of <code>FileInfo[]</code> . <u>Syntax:</u> <code>referencevariable.GetFiles()</code>
6	GetDirectories()	Returns the list of sub directories of the current folder, as an array of <code>DirectoryInfo[]</code> . <u>Syntax:</u> <code>referencevariable.GetDirectories()</code>

The “System.IO.DirectoryInfo” class - Example

Creating Project

- Create the following folders and files:
 - C:\CSharp
 - sample
 - firstfolder (folder)
 - secondfolder (folder)
 - thirdfolder (folder)
 - New Text Document.txt
 - New Microsoft Word Document.docx

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ DirectoryInfoExample ”.
- Type the location as “ C:\CSharp ”.
- Type the solution name as “ DirectoryInfoExample ”.
- Click on OK.

Program.cs

```
using System;
using System.IO;

namespace namespace1
{
    class Program
    {
        static void Main()
        {
            DirectoryInfo d;
            d = new DirectoryInfo(@"c:\CSharp\myfolder");
            Console.WriteLine("Exists: " + d.Exists);

            if (d.Exists)
            {
                Console.WriteLine("Full name: " + d.FullName);
                Console.WriteLine("Name: " + d.Name);
                Console.WriteLine("Directory name: " + d.Parent);
                Console.WriteLine("Creation date and time: " + d.CreationTime);
                Console.WriteLine("Modification date and time: " +
d.LastWriteTime);
                Console.WriteLine("Access date and time: " + d.LastAccessTime);

                Console.WriteLine("\nFiles:");
                FileInfo[] files = d.GetFiles();
                for (int i = 0; i < files.Length; i++)
                {

```

```

        Console.WriteLine(files[i].FullName);
    }

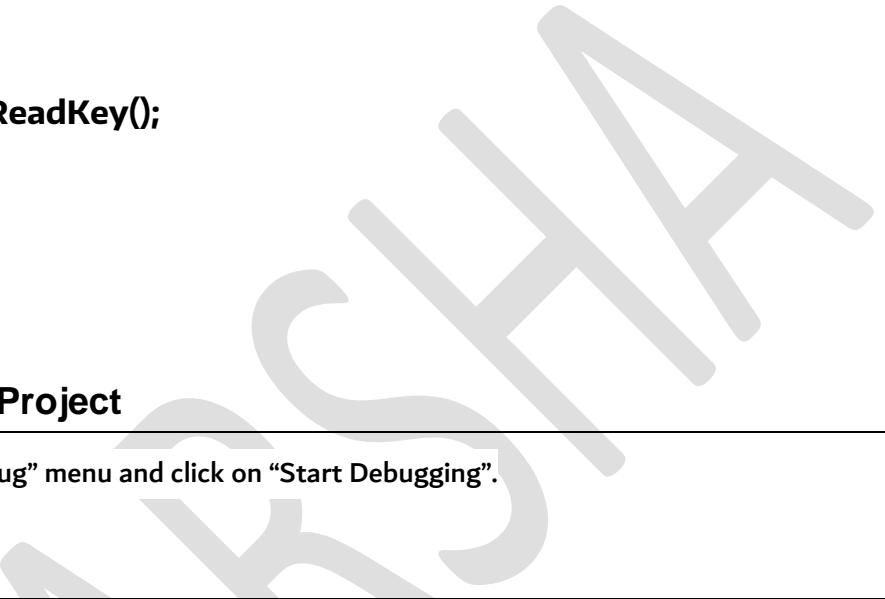
    Console.WriteLine("\nSub directories:");
    DirectoryInfo[] directories = d.GetDirectories();
    for (int i = 0; i < directories.Length; i++)
    {
        Console.WriteLine(directories[i].FullName);
    }
}
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/265. Console - IO - DirectoryInfo DirectoryInfoExample/bin/Debug/DirectoryInfoExample.E... ━ ━ X
Exists: True
Full name: c:\CSharp\myfolder
Name: myfolder
Directory name: CSharp
Creation date and time: 9/23/2016 4:37:04 PM
Modification date and time: 9/23/2016 4:37:04 PM
Access date and time: 9/23/2016 4:37:04 PM

Files:
c:\CSharp\myfolder\file1.txt
c:\CSharp\myfolder\file2.txt
c:\CSharp\myfolder\file3.txt

Sub directories:
c:\CSharp\myfolder\subfolder1
c:\CSharp\myfolder\subfolder2
c:\CSharp\myfolder\subfolder3

```

It shows the folder details as above.

The “System.IO.Directory” class

- The “Directory” is a class in “System.IO” namespace.
- The “Directory” class is a static class, which provides a set of static methods to manipulate folders.

Steps for development of “Directory” class:

- Import the namespace:
 - `using System.IO;`
- Call any static method:
 - `Directory.method();`

Methods of “Directory” class:

Sl. No	Method	Description
1	<code>Exists()</code>	<p>Checks whether the folder exists or not.</p> <p><u>True:</u> The folder exists.</p> <p><u>False:</u> The folder not exists</p> <p><u>Syntax:</u> <code>Directory.Exists(string FolderPath)</code></p>
2	<code>CreateDirectory()</code>	<p>Create the specified folder and returns corresponding <code> DirectoryInfo</code>’s object.</p> <p><u>Syntax:</u> <code>Directory.CreateDirectory(string FolderPath)</code></p>

3	Delete()	<p>Deletes the specified folder permanently, including all of its sub folders and files in it.</p> <p><u>Syntax:</u> Directory.Delete(string FolderPath, true)</p> <p><u>Warning:</u> This action is permanent. It is impossible to undo this action. Check the folder path before running the program.</p>
4	Move()	<p>Moves the specified source folder into the specified destination location.</p> <p><u>Rule:</u> Both source and destination location should be in the same drive. Ex: C: drive</p> <p><u>Syntax:</u> Directory.Move(string SourceFolderPath, string DestinationFolderPath)</p>

The “System.IO.Directory” class - Example

Creating Project

- Create the following folders:
 - C:\CSharp
 - Sample (folder)

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “DirectoryExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DirectoryExample”.
- Click on OK.

Program.cs

```
using System;
using System.IO;

namespace DirectoryExample
{
    class Program
    {
        static void Main()
        {
            if(Directory.Exists(@"c:\folder1") == true)
            {
                Directory.Delete(@"C:\folder1", true);
            }
            if(Directory.Exists(@"c:\folder2") == true)
            {
                Directory.Delete(@"C:\folder2", true);
            }

            Directory.CreateDirectory(@"C:\folder1");
            Console.WriteLine("folder1 created");
            Console.WriteLine("folder1 exists: " + Directory.Exists(@"C:\folder1"));

//Output: True

            Directory.Delete(@"C:\folder1");
            Console.WriteLine("folder1 deleted");
            Console.WriteLine("folder1 exists: " + Directory.Exists(@"C:\folder1"));

//Output: False
```

```
Directory.CreateDirectory(@"C:\folder1");
Console.WriteLine("folder1 created");

Directory.Move(@"C:\folder1", @"C:\folder2");
Console.WriteLine("folder1 moved as folder2");

Console.ReadKey();
}

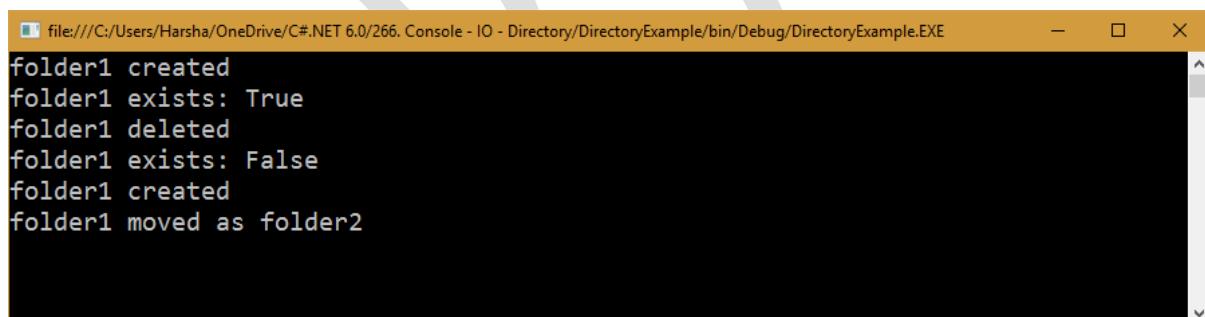
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/266. Console - IO - Directory/DirectoryExample/bin/Debug/DirectoryExample.EXE
folder1 created
folder1 exists: True
folder1 deleted
folder1 exists: False
folder1 created
folder1 moved as folder2
```

The “System.IO.File” class

System.IO.File class

- The “File” is a class in “System.IO” namespace.
- The “File” class is a static class, which provides a set of static methods to manipulate files.

Steps for development of “File” class:

- Import the namespace:
 - `using System.IO;`
- Call any static method:
 - `File.method();`

Methods of “File” class:

Sl. No	Method	Description
1	<code>Exists()</code>	<p>Checks whether the file exists or not.</p> <p><u>True:</u> The file exists.</p> <p><u>False:</u> The file not exists</p> <p><u>Syntax:</u> <code>File.Exists(string FilePath)</code></p>
2	<code>Create()</code>	<p>Create the specified file and returns corresponding <code>FileStream</code>’s object.</p> <p><u>Syntax:</u> <code>File.Create(string FilePath)</code></p>

3	Delete()	Deletes the specified file permanently. <u>Syntax:</u> File.Delete(string FilePath) Warning: This action is permanent. It is impossible to undo this action. Check the file path before running the program.
4	Move()	Moves the specified source file into the specified destination location. <u>Syntax:</u> File.Move(string SourceFilePath, string DestinationFilePath)
5	Copy()	Copies the specified source file into the specified destination location. <u>Syntax:</u> File.Copy(string SourceFilePath, string DestinationFilePath)

The “System.IO.File” class - Example

Creating Project

- Create the following folders and files:
 - C:\CSharp
- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “FileExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FileExample”. Click on OK.

Program.cs

```
using System;
using System.IO;

namespace FileExample
{
    class Program
    {
        static void Main()
        {
            if (Directory.Exists(@"c:\folder1") == true)
            {
                Directory.Delete(@"c:\folder1", true);
            }

            Directory.CreateDirectory(@"c:\folder1");
            Console.WriteLine("Folder created");

            File.Create(@"c:\folder1\file1.txt").Close();
            Console.WriteLine("File created");

            File.Copy(@"c:\folder1\file1.txt", @"c:\folder1\file2.txt");
            Console.WriteLine("File created");

            File.Move(@"c:\folder1\file2.txt", @"c:\folder1\file3.txt");
            Console.WriteLine("File moved");

            Console.WriteLine("File exists: " + File.Exists(@"c:\folder1\file3.txt"));
            Console.ReadKey();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/267. Console - IO - File/FileExample/bin/Debug/FileExample.EXE
Folder created
File created
File created
File moved
File exists: True
```

The “System.IO.FileStream” class

- The “FileStream” is a class in “System.IO” namespace.
- The “FileStream” class is used to write data to a file (or) read data from an existing file in byte[] format.
- But in fact, it is difficult to deal with byte[]. That's why, often we use StreamWriter or StreamReader to perform file reading or writing operations.
- “FileStream” is useful while working with “StreamWriter” or “StreamReader”.
- “FileStream” is a child class of a parent class called “Stream”.
- “FileStream” supports two modes majorly:
 - Create mode: Used to create a new file and write data into the new file.
 - Read mode: Used to read data from an existing file.

Steps for development of “FileStream” class (Create mode):

- Import the namespace:
 - `using System.IO;`
- Create a reference variable:
 - `FileStream referencevariable;`
- Create an object:
 - `referencevariable = new FileStream("file path", FileMode.Create, FileAccess.Write);`

Steps for development of “FileStream” class (Open mode):

- Import the namespace:
 - `using System.IO;`

- Create a reference variable:
 - `FileStream referencevariable;`

- Create an object:
 - `referencevariable = new FileStream(“file path”, FileMode.Open, FileAccess.Read);`

The “System.IO.StreamWriter” class

System.IO.StreamWriter class

- The “StreamWriter” is a class in “System.IO” namespace.
- The “StreamWriter” class is used to write data to a new / existing file in text format.
- “StreamWriter” is usually used in combination with “FileStream”.

Steps for development of “StreamWriter” class:

- Import the namespace:
 - `using System.IO;`
- Create a reference variable:
 - `StreamWriter referencevariable;`
- Create an object:
 - `referencevariable = new StreamWriter(file stream object here);`
- Write data to file:
 - `referencevariable.Write (string content);`
- Close the file:
 - `referencevariable.Close();`

The “System.IO.StreamWriter” class - Example

Creating Project

- Create the following folder:
 - C:\CSharp
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StreamWriterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StreamWriterExample”.
- Click on OK.

Program.cs

```
//city.txt
using System;
using System.IO;

namespace namespace1
{
    class Program
    {
        static void Main()
        {
            //delete the file, if it already exists
            FileInfo f = new FileInfo(@"c:\CSharp\file1.txt");
            if (f.Exists == true)
                f.Delete();

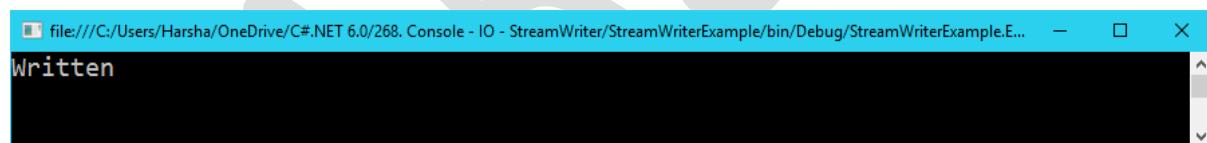
            //create the file
            FileStream fs = new FileStream(@"c:\CSharp\file1.txt",
                FileMode.Create, FileAccess.Write);
```

```
//write data to the file
StreamWriter sw = new StreamWriter(fs);
sw.WriteLine("Hai");
sw.WriteLine("Hello");
sw.WriteLine("How are you");
//close the file
sw.Close();
Console.WriteLine("Written");
Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The file will be created at “C:\Sharp\sample.txt”.

The “System.IO.StreamReader” class

System.IO.StreamReader class

- The “StreamReader” is a class in “System.IO” namespace.
- The “StreamReader” class is used to read data from an existing file in text format.
- “StreamReader” is usually used in combination with “FileStream”.

Steps for development of “StreamReader” class:

- Import the namespace:
 - `using System.IO;`
- Create a reference variable:
 - `StreamReader referencevariable,`
- Create an object:
 - `referencevariable = new StreamReader(file stream object here);`
- Read complete content of the file:
 - `string variablename = referencevariable.ReadToEnd();`
- Close the file:
 - `referencevariable.Close();`

The “System.IO.StreamReader” class - Example

Creating Project

- Create the following folder:
 - C:\CSharp
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “StreamReaderExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StreamReaderExample”.
- Click on OK.

Program.cs

```
//city.txt
using System;
using System.IO;

namespace namespace1
{
    class Program
    {
        static void Main()
        {
            //check the file exists or not
            FileInfo f = new FileInfo(@"c:\CSharp\file1.txt");
            if (f.Exists == true)
            {
                //open the file
```

```

FileStream fs = new FileStream(@"c:\CSharp\file1.txt",
 FileMode.Open, FileAccess.Read);

//read the file
StreamReader sr = new StreamReader(fs);
string s = sr.ReadToEnd();

//close the file
sr.Close();
Console.WriteLine(s);
}

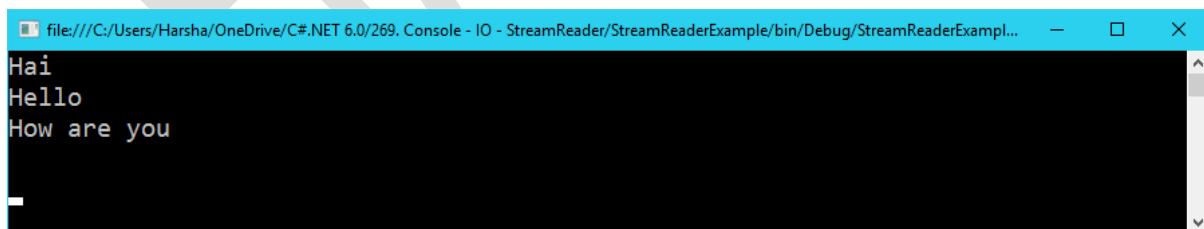
else
{
    Console.WriteLine("File not found");
}
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window titled "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/269. Console - IO - StreamReader/StreamReaderExample/bin/Debug/StreamReaderExampl...". The window displays the following text:

```
Hai
Hello
How are you
```

It shows content from “C:\CSharp\sample.txt” file.

C#.NET – Console – ADO.NET

Database Basics

- **DBMS (Database Management System)**: It is a software, which is used to store and manage databases. Ex: SQL Server, Oracle etc.
- **Database**: It is a collection of tables. Databases are used to store data permanently.
- **Table**: It is a collection of rows and columns.
- **Column**: A field in the table is called as column.
- **Row**: A record in the table is called as row.

Table Structure

Sl. No	Column Name	Data Type
1	Column1	Data type 1
2	Column2	Data type 2
3	Column3	Data type 3

Example of Table Structure: Employees

Sl. No	Column Name	Data Type
1	EmpID	Int
2	EmpName	Nvarchar(max)
3	Salary	decimal

Table: Employees

EmplID	EmpName	Salary
1	Scott	5000
2	Allen	6000
3	Jones	7000
4	John	8000
5	Mark	9000

HARSH

Introduction to ADO.NET

Introduction to ADO.NET

- ADO.NET (ActiveX Data Objects .NET) is a “database technology” or “database framework”, which is used to connect and interact with databases such as SQL Server, Oracle, Excel, Access etc.
- ADO.NET is a collection of pre-defined classes.
- ADO.NET can be used in c#.net and asp.net also.

List of pre-defined classes in ADO.NET

- ADO.NET provides the following pre-defined classes.
 1. System.Data.SqlClient.SqlConnection
 2. System.Data.SqlClient.SqlCommand
 3. System.Data.SqlClient.SqlDataReader
 4. System.Data.SqlClient.SqlDataAdapter
 5. System.Data.SqlClient.SqlParameter
 6. System.Data.SqlClient.SqlCommandBuilder
 7. System.Data.SqlClient.SqlTransaction
 8. System.Data.OleDb.OleDbConnection
 9. System.Data.OleDb.OleDbCommand
 10. System.Data.OleDb.OleDbDataReader
 11. System.Data.OleDb.OleDbDataAdapter
 12. System.Data.OleDb.OleDbParameter
 13. System.Data.OleDb.OleDbCommandBuilder
 14. System.Data.OleDb.OleDbTransaction

15. System.Data.DataSet
16. System.Data.DataTable
17. System.Data.DataRow
18. System.Data.DataColumn
19. System.Data.DataView

ADO.NET – “SqlConnection” class

System.Data.SqlClient.SqlConnection

- The “SqlConnection” is a class, which is a member of “System.Data.SqlClient” namespace.
- This class’s object represents a connection to SQL Server database.

Properties of “SqlConnection” class:

Sl. No	Property	Data Type	Description
1	ConnectionString	string	<p>Represents details about the connection, based on which ado.net should establish a connection with database.</p> <p><u>Syntax:</u> <i>referencevariable.ConnectionString</i> = “connection string here”;</p>
2	State	ConnectionState	<p>Represents current status of the connection, whether it is opened or closed. It is readonly property.</p> <p><u>Syntax:</u> <i>referencevariable.State</i></p>

Constructors of “SqlConnection” class:

Sl. No	Constructor	Description
1	<code>SqlConnection()</code>	Initializes nothing. <u>Syntax:</u> <code>new SqlConnection();</code>
2	<code>SqlConnection(string ConnectionString)</code>	Initializes “ConnectionString” property. <u>Syntax:</u> <code>new SqlConnection("connection string here");</code>

Methods of “SqlConnection” class:

Sl. No	Method	Return Data Type	Description
1	<code>Open()</code>	<code>void</code>	Opens the connection. After opening the connection, SQL Server listens the requests you made. <u>Syntax:</u> <code>referencevariable.Open()</code>
2	<code>Close()</code>	<code>void</code>	Closes the connection. After closing the connection, SQL Server stops listening the requests you made. <u>Syntax:</u> <code>referencevariable.Close()</code>

Steps for development of “SqlConnection” class:

- Import the namespace:
 - `using System.Data.SqlClient;`
- Create a reference variable:
 - `SqlConnection referencevariable;`
- Create an object:
 - `referencevariable = new SqlConnection();`
- Set connection string:
 - `referencevariable.ConnectionString = "connection string here";`
- Open the connection:
 - `referencevariable.Open();`
- Close the connection:
 - `referencevariable.Close();`

Connection Strings for SQL Server

Sl. No	Type of connection	Description	Connection String
1	Windows Authentication	The current working windows username and password will be automatically submitted to SQL Server.	"data source= <i>servernamehere</i> ; integrated security=yes; initial catalog= <i>databasenamehere</i> "
2	SQL Server Authentication	We can submit SQL Server username and password to SQL Server.	"data source= <i>servernamehere</i> ; user id= <i>usernamehere</i> ; password= <i>passwordhere</i> ; initial catalog= <i>databasenamehere</i> "

SqlConnection – Windows Authentication – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “WindowsAuthExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WindowsAuthExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace WindowsAuthExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable
            SqlConnection cn;

            //create object
            cn = new SqlConnection();

            //calling properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";

            //calling methods
            Console.WriteLine(cn.State); //Output: Closed
            cn.Open();
            Console.WriteLine(cn.State); //Output: Open
            cn.Close();
            Console.WriteLine(cn.State); //Output: Closed
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/273. Console - ADO.NET - Windows Auth/WindowsAuthExample/bin/Debug/WindowsAut...
Closed
Open
Closed
```

Note: If any database connection problem, it shows exception (run time error).

SqlConnection – SQL Server Authentication – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “SqlServerAuthExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerAuthExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace SqlServerAuthExample
{
    class Program
    {
        static void Main()
        {
            //create reference variable
            SqlConnection cn;

            //create object
            cn = new SqlConnection();

            //calling properties
            cn.ConnectionString = "data source=localhost; user id=sa;
password=123; initial catalog=company";

            //calling methods
            Console.WriteLine(cn.State); //Output: Closed
            cn.Open();
            Console.WriteLine(cn.State); //Output: Open
            cn.Close();
            Console.WriteLine(cn.State); //Output: Closed

            Console.ReadKey();
        }
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/274. Console - ADO.NET - SQL Server Auth/SqlServerAuthExample/bin/Debug/SqlServerAu... — □ ×  
Closed  
Open  
Closed  
-
```

Note: If any database connection problem, it shows exception (run time error).

The "SqlCommand" class in ADO.NET

- The “SqlCommand” is a pre-defined class in a namespace called “System.Data.SqlClient”.
- This class’s object is used to execute an SQL statement or stored procedure through database connection.
- It requires to use SqlConnection class, to connect with database.
- It requires to use SqlParameter class, to pass parameters to DBMS.

Properties of “SqlCommand” class:

Sl. No	Property	Data Type	Description
1	CommandText	string	<p>Represents the sql statement or store procedure name, which has to be executed at the DBMS.</p> <p><u>Syntax:</u> <code>referencevariable.CommandText = "sql statement stored procedure name here";</code></p>
2	Connection	SqlConnection	<p>Represents the connection object, based on which the command has to be executed.</p> <p><u>Syntax:</u> <code>referencevariable.Connection = ReferenceVariableOfSqlConnectionClass;</code></p>

3	CommandType	CommandType	<p>Represents type of the command.</p> <p><u>Options:</u> Text StoredProcedure</p> <ul style="list-style-type: none"> a) Text = Represents a command with a normal SQL statement, such as SELECT INSERT UPDATE DELETE etc. b) StoredProcedure = Represents a command with stored procedure. <p><u>Syntax:</u> <i>referencevariable.CommandType</i> = System.Data.CommandType.Optionhere;</p>
4	Parameters	List<SqlParameter>	<p>Represents the collection of parameters that are to be passed to DBMS in order to execute the command.</p> <p><u>Syntax:</u></p> <pre><i>referencevariable.Parameters.Add(ReferenceVariableOfSqlParameter)</i></pre>
5	Transaction	SqlTransaction	<p>Represents the transaction, in which the command is a part.</p> <p><u>Syntax:</u> <i>referencevariable.Transaction</i> = ReferenceVariableOfSqlTransaction;</p>

Constructors of “SqlCommand” class:

Sl. No	Constructor	Description
1	<code>SqlCommand()</code>	Initializes nothing. <u>Syntax:</u> <code>new SqlCommand();</code>
2	<code>SqlCommand(string CommandText, SqlConnection connection)</code>	Initializes “CommandText” and “Connection” properties. <u>Syntax:</u> <code>new SqlCommand("command text here", cn);</code>

Methods of “SqlCommand” class:

Sl. No	Method	Return Data Type	Description
1	<code>ExecuteScalar()</code>	<code>object</code>	<p>Executes the SELECT statement and returns only one result value.</p> <p>This method requires the connection in “Open” status.</p> <p><u>Syntax:</u> <code>referencevariable.ExecuteScalar()</code></p>
2	<code>ExecuteReader()</code>	<code>SqlDataReader</code>	<p>Executes the SELECT statement & creates and returns an object of SqlDataReader class.</p> <p>This method requires the connection in “Open” status.</p> <p><u>Syntax:</u> <code>referencevariable.ExecuteReader()</code></p>

3	ExecuteNonQuery()	int	<p>Executes the INSERT UPDATE DELETE statement & returns the no. of rows affected.</p> <p>This method requires the connection in “Open” status.</p> <p><u>Syntax:</u></p> <p><i>referencevariable.ExecuteNonQuery()</i></p>
---	-------------------	-----	---

HARSHA

ADO.NET - ExecuteScalar

ExecuteScalar

- “ExecuteScalar” method is used to execute a “SELECT statement” or “stored procedure” and get the single result value.

Steps for development of “ExecuteScalar” method:

- Import the namespace:
 - using System.Data.SqlClient;
- Create reference variables:
 - SqlConnection cn;
 - SqlCommand cmd;
- Create objects:
 - cn = new SqlConnection();
 - cmd = new SqlCommand();
- Call properties
 - cmd.CommandText = “*comamnd text here*”;
 - cmd.Connection = cn;
- Call methods
 - cn.Open();
 - object variablename = cmd.ExecuteScalar();
 - cn.Close();

SqlCommand – ExecuteScalar – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ExecuteScalarExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExecuteScalarExample”. Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace ExecuteScalarExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();

            /* call properties */
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "select EmpName from Employees where
EmpID=1";
            cmd.Connection = cn;

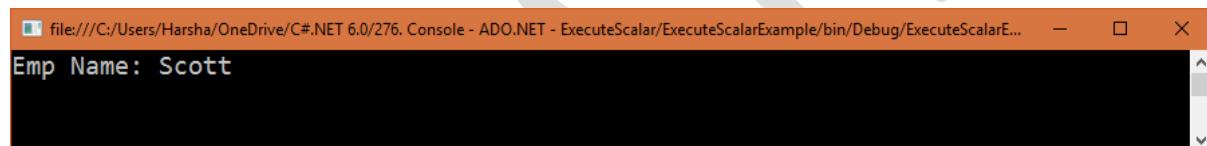
            /* call methods */
            cn.Open();
            object obj = cmd.ExecuteScalar();
        }
    }
}
```

```
        cn.Close();
        string n = Convert.ToString(obj);
        string msg = "Emp Name: " + n;
        Console.WriteLine(msg);
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Note: If any database connection problem, it shows exception (run time error).

SqlCommand – ExecuteScalar – Example 2

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “ExecuteScalarExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExecuteScalarExample2”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace ExecuteScalarExample2
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();

            /* call properties */
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "select count(*) from Employees";
            cmd.Connection = cn;

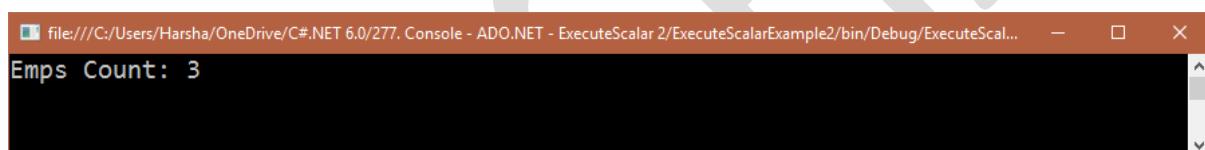
            /* call methods */
            cn.Open();
            object obj = cmd.ExecuteScalar();
        }
    }
}
```

```
cn.Close();
string n = Convert.ToString(obj);
Console.WriteLine("Emps Count: " + n);
Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Note: If any database connection problem, it shows exception (run time error).

ADO.NET – Connection Oriented Model – Introduction

- The “ADO.NET Connection Oriented Model” concept is used to read the data from database in “record-by-record” (row-by-row) approach.
- ADO.NET sends request to the database server and gets only one record at-a-time. It stores the current record in an object of “SqlDataReader” class.
- The connection should be in “Open” status, while retrieving data from database.

Advantages of Connection Oriented Model:

- At-a-time ONLY ONE record will be stored in .net application memory (RAM). So it requires less amount of memory.

Dis-advantages of Connection Oriented Model:

- It is a bit slow-process because of “record-by-record” approach.
- It supports “sequential retrieval of records”. We can’t retrieve a record based on its index. We can’t retrieve records randomly or in reverse order.

List of classes required for ADO.NET Connection Oriented Model:

1. System.Data.SqlClient.SqlConnection
 - Used to connect with SQL Server.
2. System.Data.SqlClient.SqlCommand
 - Used to send and execute an SQL statement to SQL Server.
3. System.Data.SqlClient.SqlDataReader
 - Used to read records one-by-one.

The "SqlDataReader" class in ADO.NET

- The “SqlDataReader” is a pre-defined class in a namespace called “System.Data.SqlClient”.
- This class’s object is used to store a single record, while retrieving data from database in “ADO.NET Connection Oriented Model”.
- It occupies the memory, which is enough for storing single database record.
- “SqlDataReader” is useful only in “ADO.NET Connection Oriented Model”.

Properties of “SqlDataReader” class:

Sl. No	Property	Data Type	Description
1	[column index]	object	<p>Retrieves the column value in the current record, based on the column index.</p> <p><u>Syntax:</u> <code>referencevariable [column index]</code></p>
2	["column name"]	object	<p>Retrieves the column value in the current record, based on the column name.</p> <p><u>Syntax:</u> <code>referencevariable ["column name"]</code></p>
3	FieldCount	int	<p>Represents the no. of fields (columns) in the current record.</p> <p><u>Syntax:</u> <code>referencevariable.FieldCount</code></p>
4	HasRows	bool	<p>Represents a Boolean value, whether the datareader object has any rows or not.</p> <p><u>Syntax:</u> <code>referencevariable.HasRows</code></p>

Constructors of “SqlDataReader” class:

Sl. No	Constructor	Description
No constructors		

Methods of “SqlDataReader” class:

Sl. No	Method	Return Data Type	Description
1	Read()	bool	<p>Reads the next record into the DataReader.</p> <p>It returns “true”, if data found.</p> <p>It returns “false”, if it is reached to end of records.</p> <p><u>Syntax:</u> <i>referencevariable.Read()</i></p>
2	GetValue(int columnindex)	object	<p>Returns the value of the specified column in the current record.</p> <p><u>Syntax:</u> <i>referencevariable.GetValue(int columnindex)</i></p>

ADO.NET – Connection Oriented Model

Steps for development of “ADO.NET Connection Oriented Model”:

- Import the namespace:
 - `using System.Data.SqlClient;`
- Create reference variables:
 - `SqlConnection cn;`
 - `SqlCommand cmd;`
 - `SqlDataReader dr;`
- Create objects:
 - `cn = new SqlConnection();`
 - `cmd = new SqlCommand();`
- Call properties
 - `cmd.CommandText = "comamnd text here";`
 - `cmd.Connection = cn;`
- Call methods
 - `cn.Open();`
 - `dr = cmd.ExecuteReader();`
 - `dr.Read();`

Note: We are creating an object for “SqlDataReader” class; because it will be created automatically when we call `ExecuteReader()` method.

- dr["column name"];
- cn.Close();

ADO.NET Connection Oriented Model – Single Record – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go

create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go

insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “COMSingleRecordExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMSingleRecordExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace COMSingleRecordExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();

            /* call properties */
        }
    }
}
```

```

cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees where empid=1";
cmd.Connection = cn;

/* call methods */
cn.Open();
dr = cmd.ExecuteReader();
if (dr.Read())
{
    object obj1, obj2, obj3;

    obj1 = dr["EmpID"];
    obj2 = dr["EmpName"];
    obj3 = dr["Salary"];
    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    Console.WriteLine("EmpID: " + eid);
    Console.WriteLine("EmpName: " + ename);
    Console.WriteLine("Salary: " + sal);
}

cn.Close();
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/280. Console - ADO.NET COM - Single Record/COMSingleRecordExample/bin/Debug/CO...
EmpID: 1
EmpName: Scott
Salary: 4000
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

ADO.NET Connection Oriented Model – Multiple Records – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “COMMmultipleRecordsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMMmultipleRecordsExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace COMMmultipleRecordsExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();

            /* call properties */
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "select * from Employees";
            cmd.Connection = cn;
        }
    }
}
```

```
/* call methods */
cn.Open();
dr = cmd.ExecuteReader();
while (dr.Read())
{
    object obj1, obj2, obj3;
    obj1 = dr["EmpID"];
    obj2 = dr["EmpName"];
    obj3 = dr["Salary"];
    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    Console.WriteLine("EmpID: " + eid + "\nEmpName: " + ename +
"\nSalary: " + sal + "\n");
}
cn.Close();
Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/281. Console - ADO.NET COM - Multiple Records/COMMultipleRecordsExample/bin/Debug/
```

```
EmpID: 1
EmpName: Scott
Salary: 4000

EmpID: 2
EmpName: Allen
Salary: 5000

EmpID: 3
EmpName: Jones
Salary: 6000
```

Note: If any database connection problem, it shows exception (run time error).

The "SqlParameter" class in ADO.NET

System.Data.SqlClient.SqlParameter

- The “SqlParameter” is a pre-defined class in a namespace called “System.Data.SqlClient”.
- This class's object represents a parameter (name and value), that is to be passed to DBMS for execution of command.
- It is useful all over in ADO.NET, such as ExecuteScalar, Connection Oriented Model, Disconnected Model, Non Query Operations, Stored Procedures, Transactions etc.

Properties of “SqlParameter” class:

Sl. No	Property	Data Type	Description
1	ParameterName	string	<p>Represents name of the parameter.</p> <p><u>Syntax:</u> <code>referencevariable.ParameterName = "parameter name here"</code></p>
2	Value	object	<p>Represents the actual value of the parameter.</p> <p><u>Syntax:</u> <code>referencevariable.Value</code></p>
3	DbType	DbType	<p>Represents the database data type of the parameter.</p> <p><u>Syntax:</u> <code>referencevariable.DbType = DbType.Optionhere;</code></p>

4	Direction	ParameterDirection	<p>Represents direction of the parameter, whether it has to be given to DBMS or has to be retrieved from DBMS.</p> <p><u>Options:</u> Input Output InputOutput ReturnValue</p> <p><u>Syntax:</u> <i>referencevariable.Direction = ParameterDirection.Optionhere;</i></p>
---	-----------	--------------------	--

Constructors of “SqlParameter” class:

Sl. No	Constructor	Description
1	SqlParameter()	<p>Initializes nothing.</p> <p><u>Syntax:</u> new SqlParameter();</p>
2	SqlParameter(string ParameterName, object Value)	<p>Initializes ParameterName and Value properties.</p> <p><u>Syntax:</u> new SqlParameter("parameter name here", value here);</p>

Steps for development of “SqlParameter”:

- Import the namespace:
 - `using System.Data.SqlClient;`

- Create reference variable:
 - `SqlParameter p;`

- Create object:
 - `p = new SqlParameter();`
- Call properties
 - `p.ParameterName = "parameter name here";`
 - `p.Value = "value here";`
- Add parameter to command:
 - `cmd.Parameters.Add(p);`

ADO.NET Connection Oriented Model – SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “COMSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMSqlParameterExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace COMSqlParameterExample
{
    class Program
    {
        static void Main()
        {
            //get empid from keyboard
            Console.Write("Emp ID: ");
            int n = Convert.ToInt32(Console.ReadLine());

            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1;
            SqlDataReader dr;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
```

```
/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees where
empid=@EmpID";
cmd.Connection = cn;
p1.ParameterName = "@EmpID";
p1.Value = n;
cmd.Parameters.Add(p1);

/* call methods */
cn.Open();
dr = cmd.ExecuteReader();
if (dr.Read())
{
    object obj1, obj2;

    obj1 = dr["EmpName"];
    obj2 = dr["Salary"];

    string ename;
    decimal sal;

    ename = Convert.ToString(obj1);
    sal = Convert.ToDecimal(obj2);

    Console.WriteLine("Emp Name: " + ename);
    Console.WriteLine("Salary: " + sal);
}
else
{
    Console.WriteLine("No data found");
}
cn.Close();
Console.ReadKey();
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/283. Console - ADO.NET COM - SqlParameter/COMSqlParameterExample/bin/Debug/CO...
Emp ID: 1
Emp Name: Scott
Salary: 4000
```

Enter EmpID as “1” and press Enter.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model

- The “System.Data.SqlClient.SqlDataAdapter” class’s object is used to convert the data from “result-set” format to “data-set” format.
 - Resultset = The data (rows and columns) that is returned from SQL server, after execution of a SELECT statement.
 - DataSet = The data that is stored in .net application memory (RAM) in the form of objects.
- It loads all records at-once.
- “ADO.NET Disconnected Model” is a concept to retrieve data from database. It is implemented using “SqlDataAdapter”.
- After loading all records at-once, the connection can be closed. That’s why it is called “ADO.NET Disconnected Model”.

Advantages of Disconnected Model:

- At-a-time ALL records will be stored (loaded) into .net application memory (RAM). After that you can disconnect from database. So your app runs faster.
- We can retrieve a record based on the index.

Dis-advantages of Disconnected Model:

- ALL records need to be stored in RAM, up to end of the .net application, so it may be burden on the RAM, if there are so many records.

The “SqlDataAdapter” class

The “SqlDataAdapter” class has following members:

Properties

- **SqlCommand SelectCommand:** It represents the object of SqlCommand class, based on which the data is to be retrieved from database.

Methods

- **Fill(DataSet dataset):** It executes the SELECT statement, converts the resultset into dataset and also stores the data in dataset.
- **Update(DataSet dataset):** It updates the changes made in the dataset to the database.

Constructors

- **SqlDataAdapter():** It initializes nothing.
- **SqlDataAdapter(SqlCommand SelectCommand):** It initializes SelectCommand property.

The “DataSet” class

- DataSet is used in ADO.NET Disconnected Model.
- DataSet temporarily stores the data that is retrieved from database.
- DataSet internally uses XML.
- DataSet can contain multiple tables.
- Every table in DataSet is treated as an object of “System.Data.DataTable” class.
- Every column in DataTable is treated as an object of “System.Data.DataColumn” class.
- Every row in DataTable is treated as an object of “System.Data.DataRow” class.

DataSet - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DataSetExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetExample”.
- Click on OK.

Program.cs

```
/* Expected data:
```

Categories:

CategoryID	CategoryName
1	ab
2	cd

Products:

ProductID	ProductName	Cost
101	prod1	400
102	prd2	500
103	prd3	600

```
*/
```

```
using System;
using System.Data;

namespace DataSetExample
{
    class Program
    {
        static void Main()
        {
            //creating dataset
            DataSet ds = new DataSet();

            //creating 2 tables
            DataTable dt1 = new DataTable() { TableName = "Categories" };
            DataTable dt2 = new DataTable() { TableName = "Products" };

            //creating 2 columns for table1
            DataColumn col1 = new DataColumn() { ColumnName = "CategoryID",
            DataType = typeof(int) };
            DataColumn col2 = new DataColumn() { ColumnName =
            "CategoryName", DataType = typeof(string) };

```

```
//creating 3 columns for table2
    DataColumn col3 = new DataColumn() { ColumnName = "ProductID",
    DataType = typeof(int) };
    DataColumn col4 = new DataColumn() { ColumnName =
    "ProductName", DataType = typeof(string) };
    DataColumn col5 = new DataColumn() { ColumnName = "Cost",
    DataType = typeof(decimal) };

//adding columns to table1
dt1.Columns.Add(col1);
dt1.Columns.Add(col2);

//adding columns to table2
dt2.Columns.Add(col3);
dt2.Columns.Add(col4);
dt2.Columns.Add(col5);

//creating 2 rows for table1
DataRow drow1 = dt1.NewRow();
DataRow drow2 = dt1.NewRow();

//creating 3 rows for table2
DataRow drow3 = dt2.NewRow();
DataRow drow4 = dt2.NewRow();
DataRow drow5 = dt2.NewRow();

//adding rows to table1
dt1.Rows.Add(drow1);
dt1.Rows.Add(drow2);

//adding rows to table2
dt2.Rows.Add(drow3);
dt2.Rows.Add(drow4);
dt2.Rows.Add(drow5);

//adding tables to dataset
```

```
ds.Tables.Add(dt1);
ds.Tables.Add(dt2);

/********** setting data *****/
//setting data in table1
dt1.Rows[0]["CategoryID"] = 1;
dt1.Rows[0]["CategoryName"] = "ab";
dt1.Rows[1]["CategoryID"] = 2;
dt1.Rows[1]["CategoryName"] = "cd";

//setting data in table2
dt2.Rows[0]["ProductID"] = 101;
dt2.Rows[0]["ProductName"] = "prod1";
dt2.Rows[0]["Cost"] = 400;
dt2.Rows[1]["ProductID"] = 102;
dt2.Rows[1]["ProductName"] = "prod2";
dt2.Rows[1]["Cost"] = 500;
dt2.Rows[2]["ProductID"] = 103;
dt2.Rows[2]["ProductName"] = "prod3";
dt2.Rows[2]["Cost"] = 600;

//getting data from table1
Console.WriteLine(dt1.TableName + ":");
for (int i = 0; i < dt1.Rows.Count; i++)
{
    Console.WriteLine(dt1.Rows[i]["CategoryID"] + ", " +
dt1.Rows[i]["CategoryName"]);
}
Console.WriteLine();

//getting data from table2
Console.WriteLine(dt2.TableName + ":");
for (int i = 0; i < dt2.Rows.Count; i++)
{
```

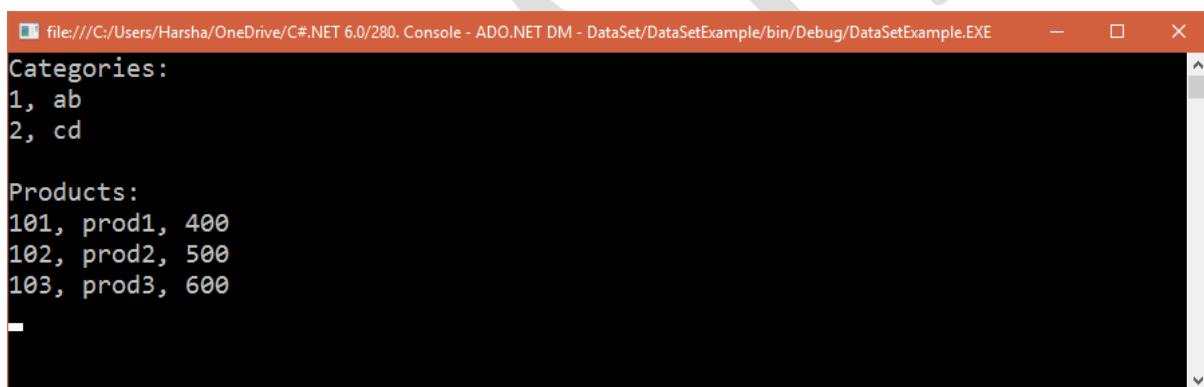
```
Console.WriteLine(dt2.Rows[i]["ProductID"] + ", " +
dt2.Rows[i]["ProductName"] + ", " + dt2.Rows[i]["Cost"]);
}

Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/280. Console - ADO.NET DM - DataSet/DataSetExample/bin/Debug/DataSetExample.EXE
Categories:
1, ab
2, cd

Products:
101, prod1, 400
102, prod2, 500
103, prod3, 600
-
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “DisconnectedModelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DisconnectedModelExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace DisconnectedModelExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();
            adp = new SqlDataAdapter();
            ds = new DataSet();

            /* call properties */
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
        }
    }
}
```

```
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);

    Console.WriteLine(eid + ", " + ename + ", " + sal);
}

Console.ReadKey();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/287. Console - ADO.NET DM - Example/DisconnectedModelExample/bin/Debug/Disconne... — X
1, Scott, 4000
2, Allen, 5000
3, Jones, 6000
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

ADO.NET Disconnected Model – Multiple Tables - Example

Creating Database

- Note: Ignore this step, if you have created “departmentsandemployeesdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database departmentsandemployeesdatabase  
go
```

```
use departmentsandemployeesdatabase  
go
```

```
create table Departments(  
DeptNo int primary key,  
DeptName nvarchar(max),  
Loc nvarchar(max))  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName varchar(max),  
Salary decimal,  
DeptNo int references Departments(DeptNo))  
go
```

```
insert into Departments values(10, 'Accounting', 'New York')  
insert into Departments values(20, 'Operations', 'New Delhi')  
insert into Departments values(30, 'Sales', 'New Jersey')
```

```
insert into Employees values(1, 'Scott', 3000, 10)  
insert into Employees values(2, 'Allen', 6500, 10)
```

```
insert into Employees values(3, 'Jones', 4577, 20)
insert into Employees values(4, 'James', 9500, 20)
insert into Employees values(5, 'Smith', 3345, 30)
insert into Employees values(6, 'Harry', 2500, 30)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MultipleTablesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultipleTablesExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace MultipleTablesExample
{
    class Program
    {
        static void Main()
```

```
{  
    /* create reference variables */  
    SqlConnection cn;  
    SqlCommand cmd;  
    SqlDataAdapter adp;  
    DataSet ds;  
    DataTable dt1, dt2;  
    DataRow drow;  
  
    /* create objects */  
    cn = new SqlConnection();  
    cmd = new SqlCommand();  
    adp = new SqlDataAdapter();  
    ds = new DataSet();  
  
    /* call properties */  
    cn.ConnectionString = "data source=localhost; integrated  
security=yes; initial catalog=departmentsandemployeesdatabase";  
    cmd.CommandText = "select * from Departments select * from  
Employees";  
    cmd.Connection = cn;  
    adp.SelectCommand = cmd;  
  
    /* call methods */  
    adp.Fill(ds);  
  
    /* departments */  
    Console.WriteLine("Departments:");  
    dt1 = ds.Tables[0];  
    int n = 50;  
    for (int i = 0; i < dt1.Rows.Count; i++)  
    {  
        drow = dt1.Rows[i];  
  
        object obj1, obj2, obj3;  
        obj1 = drow["DeptNo"];  
        obj2 = drow["DeptName"];  
    }  
}
```

```
obj3 = drow["Loc"];

int dno;
string dname;
string loc;

dno = Convert.ToInt32(obj1);
dname = Convert.ToString(obj2);
loc = Convert.ToString(obj3);

Console.WriteLine(dno + ", " + dname + ", " + loc);
}

/* employees */
Console.WriteLine("\nEmployees:");
dt2 = ds.Tables[1];
n += 100;
for (int i = 0; i < dt2.Rows.Count; i++)
{
    drow = dt2.Rows[i];

    object obj1, obj2, obj3, obj4;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    obj4 = drow["DeptNo"];

    int eid;
    string ename;
    decimal sal;
    int dno;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    dno = Convert.ToInt32(obj4);
```

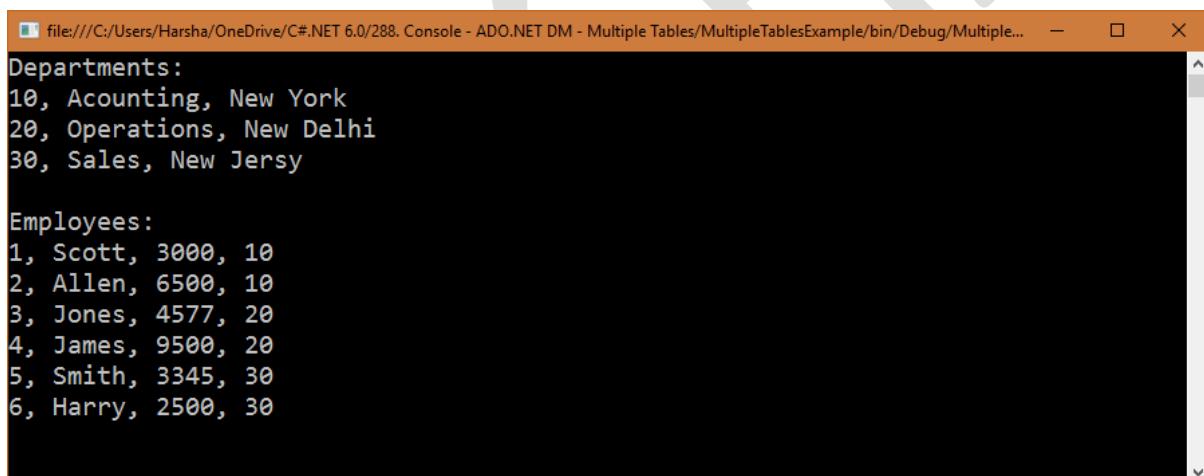
```
Console.WriteLine(eid + "," + ename + "," + sal + "," + dno);
}

Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/288. Console - ADO.NET DM - Multiple Tables/MultipleTablesExample/bin/Debug/Multiple...
Departments:
10, Accounting, New York
20, Operations, New Delhi
30, Sales, New Jersey

Employees:
1, Scott, 3000, 10
2, Allen, 6500, 10
3, Jones, 4577, 20
4, James, 9500, 20
5, Smith, 3345, 30
6, Harry, 2500, 30
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Joins - Example

Creating Database

- Note: Ignore this step, if you have created “departmentsandemployeesdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database departmentsandemployeesdatabase  
go
```

```
use departmentsandemployeesdatabase  
go
```

```
create table Departments(  
DeptNo int primary key,  
DeptName nvarchar(max),  
Loc nvarchar(max))  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName varchar(max),  
Salary decimal,  
DeptNo int references Departments(DeptNo))  
go
```

```
insert into Departments values(10, 'Acounting', 'New York')  
insert into Departments values(20, 'Operations', 'New Delhi')  
insert into Departments values(30, 'Sales', 'New Jersy')
```

```
insert into Employees values(1, 'Scott', 3000, 10)  
insert into Employees values(2, 'Allen', 6500, 10)  
insert into Employees values(3, 'Jones', 4577, 20)
```

```
insert into Employees values(4, 'James', 9500, 20)
insert into Employees values(5, 'Smith', 3345, 30)
insert into Employees values(6, 'Harry', 2500, 30)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “JoinsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “JoinsExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace JoinsExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
```

```
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
DataSet ds;
DataTable dt;
DataRow drow;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=departmentsandemployeesdatabase";
cmd.CommandText = "select * from employees inner join departments on employees.deptno=departments.deptno";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3, obj4, obj5, obj6;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    obj4 = drow["DeptNo"];
    obj5 = drow["DeptName"];
    obj6 = drow["Loc"];
    int eid;
    string ename;
```

```

decimal sal;
int dno;
string dname;
string loc;
eid = Convert.ToInt32(obj1);
ename = Convert.ToString(obj2);
sal = Convert.ToDecimal(obj3);
dno = Convert.ToInt32(obj4);
dname = Convert.ToString(obj5);
loc = Convert.ToString(obj6);
Console.WriteLine(eid + "," + ename + "," + sal + "," + dno + "," + loc);
}
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```

1, Scott, 3000, 10, New York
2, Allen, 6500, 10, New York
3, Jones, 4577, 20, New Delhi
4, James, 9500, 20, New Delhi
5, Smith, 3345, 30, New Jersy
6, Harry, 2500, 30, New Jersy

```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query - Insertion – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InsertionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InsertionExample”. Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace InsertionExample
{
    class Program
    {
        static void Main()
        {
            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();

            //calling properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "insert into Employees values(10, 'qwerty',
4500)";
            cmd.Connection = cn;
        }
    }
}
```

```
//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

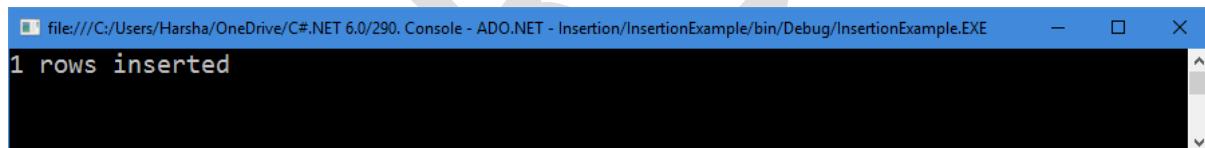
Console.WriteLine(n + " rows inserted");
Console.ReadKey();
}

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/290. Console - ADO.NET - Insertion/InsertionExample/bin/Debug/InsertionExample.EXE
1 rows inserted
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non-Query - Updation – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “UpdationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace UpdationExample
{
    class Program
    {
        static void Main()
        {
            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();

            //calling properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "update Employees set EmpName='asdf',
Salary=8900 where EmpID=1";
            cmd.Connection = cn;

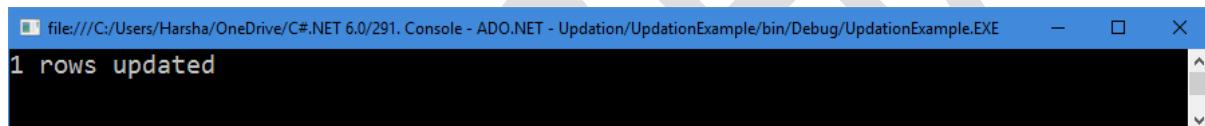
            //calling methods
            cn.Open();
            int n = cmd.ExecuteNonQuery();
            cn.Close();
        }
    }
}
```

```
Console.WriteLine(n + " rows updated");
Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/291. Console - ADO.NET - Updation/UpdationExample/bin/Debug/UpdationExample.EXE
1 rows updated
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non-Query - Deletion – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “DeletionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DeletionExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace DeletionExample
{
    class Program
    {
        static void Main(string[] args)
        {
            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();

            //calling properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "delete from Employees where EmpID=3";
            cmd.Connection = cn;

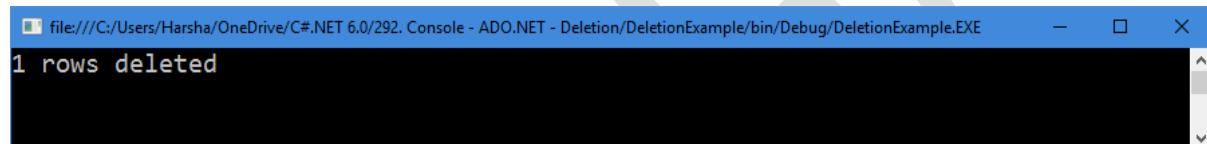
            //calling methods
            cn.Open();
            int n = cmd.ExecuteNonQuery();
            cn.Close();
        }
    }
}
```

```
        Console.WriteLine(n + " rows deleted");
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Insertion – With SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InsertionSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InsertionSqlParameterExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace InsertionSqlParameterExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());

            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1, p2, p3;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();
```

```

p1 = new SqlParameter();
p2 = new SqlParameter();
p3 = new SqlParameter();

//calling properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "insert into Employees values(@empid,
@empname, @salary)";
cmd.Connection = cn;
p1.ParameterName = "@empid";
p2.ParameterName = "@empname";
p3.ParameterName = "@salary";
p1.Value = empId;
p2.Value = empName;
p3.Value = sal;
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);

//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

Console.WriteLine(n + " rows inserted");
Console.ReadKey();
}

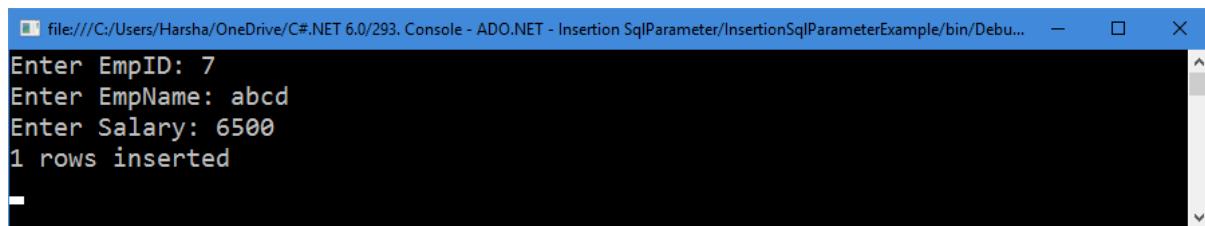
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/293. Console - ADO.NET - Insertion SqlParameter/InsertionSqlParameterExample/bin/Debug/
```

```
Enter EmpID: 7
Enter EmpName: abcd
Enter Salary: 6500
1 rows inserted
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

ADO.NET Non-Query – Updation – With SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “UpdationSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationSqlParameterExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace UpdationSqlParameterExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());
            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1, p2, p3;
            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
            p2 = new SqlParameter();
```

```
p3 = new SqlParameter();
//calling properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "update Employees set empname=@empname,
salary=@salary where empid=@empid";
cmd.Connection = cn;
p1.ParameterName = "@empid";
p2.ParameterName = "@empname";
p3.ParameterName = "@salary";
p1.Value = empId;
p2.Value = empName;
p3.Value = sal;
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);
//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();
Console.WriteLine(n + " rows updated");
Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/294. Console - ADO.NET - Updation SqlParameter/UpdationSqlParameterExample/bin/Debug/UpdationSqlParameterExample.exe

Enter Existing EmpID: 7
Enter EmpName: xyz
Enter Salary: 4455
1 rows updated
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non-Query – Deletion – With SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DeletionSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DeletionSqlParameterExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace DeletionSqlParameterExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID:");
            int empId = Convert.ToInt32(Console.ReadLine());

            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1;

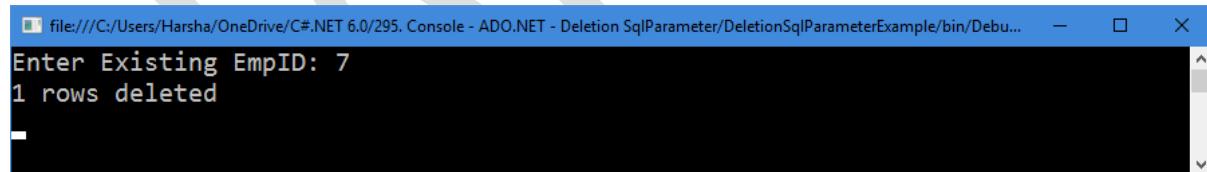
            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
            //calling properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
```

```
cmd.CommandText = "delete from Employees where empid=@empid";
cmd.Connection = cn;
p1.ParameterName = "@empid";
p1.Value = empId;
cmd.Parameters.Add(p1);
//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();
Console.WriteLine(n + " rows deleted");
Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/295. Console - ADO.NET - Deletion SqlParameter/DeletionSqlParameterExample/bin/Debu...
Enter Existing EmpID: 7
1 rows deleted
```

Note: If any database connection problem, it shows exception (run time error).

Stored Procedures Calling in ADO.NET

Calling Stored Procedures using ADO.NET

- Stored procedure is a collection of SQL statements that will be stored in database.
- Stored procedures can be called from .net application.
- Stored procedures will be compiled once and executes every time when we call it. This is called “pre-compilation”.
- Stored procedures improve performance because of pre-compilation.
- Stored procedures support to separate work between database developer and .net developer in real time.
- Stored procedures are best for performing multiple and complex database operations with a single database call.

Syntax of creating stored procedure in SQL Server

```
create procedure procedurename
(@parameter1 datatype, @parameter2 datatype, ...)
as begin
    code here
end
go
```

ADO.NET Non Query – Insertion – With Stored Procedures – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company  
go
```

```
use company  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

```
create procedure InsertEmployee(  
@EmpID int,  
@EmpName nvarchar(max),  
@Salary decimal  
)  
as begin  
    insert into Employees values(@EmpID, @EmpName, @Salary)  
end  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “InsertionWithSPExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InsertionWithSPExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace InsertionWithSPExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
```

```
Console.WriteLine("Enter Salary:");
decimal sal = Convert.ToDecimal(Console.ReadLine());

//create reference variables
SqlConnection cn;
SqlCommand cmd;
SqlParameter p1, p2, p3;

//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
p1 = new SqlParameter();
p2 = new SqlParameter();
p3 = new SqlParameter();

//calling properties
cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
cmd.CommandText = "InsertEmployee";
cmd.Connection = cn;
cmd.CommandType = CommandType.StoredProcedure;
p1.ParameterName = "@empid";
p2.ParameterName = "@empname";
p3.ParameterName = "@salary";
p1.Value = empId;
p2.Value = empName;
p3.Value = sal;
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);

//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

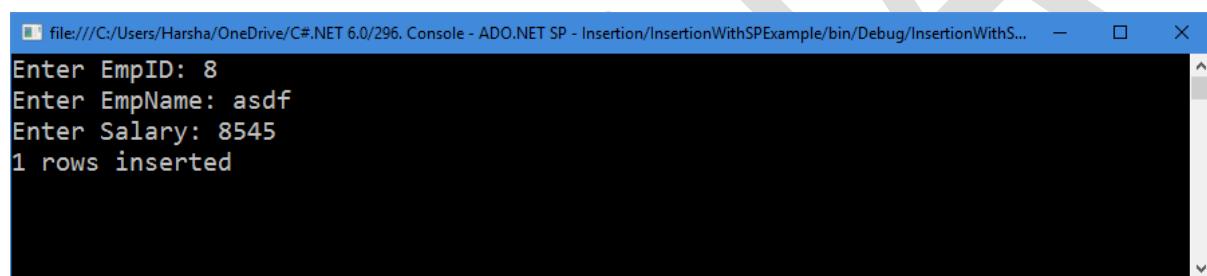
Console.WriteLine(n + " rows inserted");
```

```
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/296. Console - ADO.NET SP - Insertion/InsertionWithSPEExample/bin/Debug/InsertionWithS...
Enter EmpID: 8
Enter EmpName: asdf
Enter Salary: 8545
1 rows inserted
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Updation – With Stored Procedures – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
```

```
use company
go
```

```
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

```
create procedure UpdateEmployee(
@EmpID int,
@EmpName nvarchar(max),
@Salary decimal
<>)
```

as begin

```
update Employees set EmpName=@EmpName, Salary=@Salary
where EmpID=@EmpID
```

end

go

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “UpdationWithSPExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationWithSPExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace UpdationWithSPExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
```

```
string empName = Console.ReadLine();
Console.Write("Enter Salary: ");
decimal sal = Convert.ToDecimal(Console.ReadLine());

//create reference variables
SqlConnection cn;
SqlCommand cmd;
SqlParameter p1, p2, p3;

//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
p1 = new SqlParameter();
p2 = new SqlParameter();
p3 = new SqlParameter();

//calling properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "UpdateEmployee";
cmd.Connection = cn;
cmd.CommandType = CommandType.StoredProcedure;
p1.ParameterName = "@empid";
p2.ParameterName = "@empname";
p3.ParameterName = "@salary";
p1.Value = empId;
p2.Value = empName;
p3.Value = sal;
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);

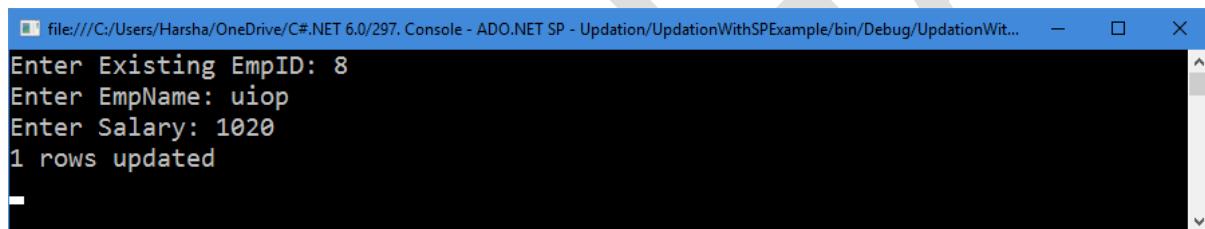
//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();
```

```
        Console.WriteLine(n + " rows updated");
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/297. Console - ADO.NET SP - Updation/UpdationWithSPExample/bin/Debug/UpdationWi...
Enter Existing EmpID: 8
Enter EmpName: uiop
Enter Salary: 1020
1 rows updated
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Deletion – With Stored Procedures – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company  
go
```

```
use company  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

```
create procedure DeleteEmployee(  
    @EmpID int  
)  
as begin  
    delete from Employees  
    where EmpID=@EmpID  
end  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “DeletionWithSPExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DeletionWithSPExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace DeletionWithSPExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());

            //create reference variables
```

```
SqlConnection cn;
SqlCommand cmd;
SqlParameter p1;

//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
p1 = new SqlParameter();

//calling properties
cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
cmd.CommandText = "DeleteEmployee";
cmd.Connection = cn;
cmd.CommandType = CommandType.StoredProcedure;
p1.ParameterName = "@empid";
p1.Value = empld;
cmd.Parameters.Add(p1);

//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

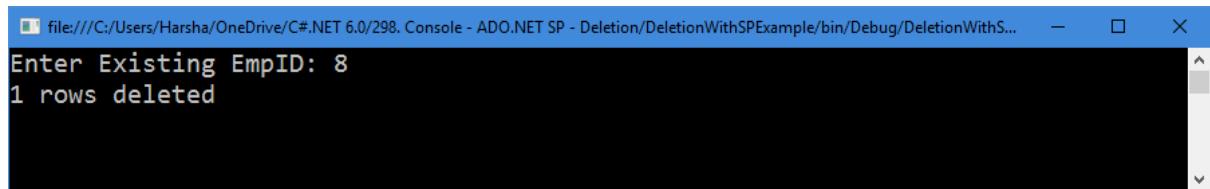
Console.WriteLine(n + " rows deleted");
Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/298. Console - ADO.NET SP - Deletion/DeletionWithSPExample/bin/Debug/DeletionWithS...
Enter Existing EmpID: 8
1 rows deleted
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

ADO.NET - Transactions

ADO.NET Transactions

- A transaction is a “collection of database operations such as insertion, deletion and updation”.
- ADO.NET transactions are used to roll back the previously executed database operations, when any database operation is failed in a transaction.
- Example: Funds transfer from one bank account to another bank account.
 - Operation 1: Debit the money from source account.
 - Operation 2: Credit the money to destination account.
- We use “System.Data.SqlClient.SqlTransaction” class is used to implement ado.net transactions in the program.

Methods for ADO.NET Transactions:

1. `cn.BeginTransaction()`
 - This method creates and starts a new ado.net transaction. It creates and returns an object for “SqlTransaction” class.
2. `transactionReferenceVariable.Commit()`
 - This method will save (fix) the database operations that are executed during the current transaction.
3. `transactionReferenceVariable.Rollback()`
 - This method will rollback (cancel) all the previously executed database operations during the current transaction.

ADO.NET – Transactions - Example

Creating Database

- Note: Ignore this step, if you have created “transactionsdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database transactionsdatabase
```

```
go
```

```
use transactionsdatabase
```

```
go
```

```
CREATE TABLE AccountsTable(
```

```
    AccountNo int primary key,
    AccountHolderName nvarchar(40),
    Balance decimal)
```

```
GO
```

```
INSERT INTO AccountsTable VALUES (101, 'scott', 10000)
```

```
INSERT INTO AccountsTable VALUES (102, 'allen', 5000)
```

```
GO
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “TransactionsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TransactionsExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;

namespace TransactionsExample
{
    class Program
    {
        static void Main()
        {
            //create reference variables
            SqlConnection cn;
            SqlTransaction transaction;
            SqlCommand cmd1, cmd2;

            //create objects
            cn = new SqlConnection();
            cmd1 = new SqlCommand();
            cmd2 = new SqlCommand();

            //call properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=transactionsdatabase";
            cmd1.CommandText = "update AccountsTable set Balance=Balance-
1000 where AccountNo=101";
            cmd2.CommandText = "update AccountsTable set
Balance=Balance+1000 were AccountNo=102";
            cmd1.Connection = cn;
```

```
cmd2.Connection = cn;

//call methods
cn.Open();
transaction = cn.BeginTransaction();
cmd1.Transaction = transaction;
cmd2.Transaction = transaction;

try
{
    cmd1.ExecuteNonQuery();
    Console.WriteLine("First operation done.");

    cmd2.ExecuteNonQuery();
    Console.WriteLine("Second operation done.");
    transaction.Commit(); //save data

    Console.WriteLine("Transaction Complete");
}
catch (Exception)
{
    transaction.Rollback(); //first operation will be rollback
    Console.WriteLine("Rollback done!");
}

cn.Close();
Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/299. Console - ADO.NET - Transactions/TransactionsExample/bin/Debug/TransactionsExa...
First operation done.
Rollback done!
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

The "OleDb" namespace in ADO.NET

System.Data.OleDb (Object Linking and Embedding Database)

- The “System.Data.OleDb” namespace contains classes like OleDbConnection, OleDbCommand etc. The usage of these classes are same as SqlConnection, SqlCommand etc.
- The System.Data.OleDb namespace is used to connect and interact with various databases like Oracle, Excel, Access etc.

Classes in “System.Data.OleDb” namespace

Sl. No	SqlClient namespace (For SQL Server)	“OleDb” namespace (For Oracle, Excel, Access)
1	System.Data.SqlClient.SqlConnection	System.Data.OleDb.OleDbConnection
2	System.Data.SqlClient.SqlCommand	System.Data.OleDb.OleDbCommand
3	System.Data.SqlClient.SqlDataReader	System.Data.OleDb.OleDbDataReader
4	System.Data.SqlClient.SqlParameter	System.Data.OleDb.OleDbParameter
5	System.Data.SqlClient.SqlDataAdapter	System.Data.OleDb.OleDbDataAdapter
6	System.Data.SqlClient.SqlCommandBuilder	System.Data.OleDb.OleDbCommandBuilder
7	System.Data.SqlClient.SqlTransaction	System.Data.OleDb.OleDbTransaction
8	System.Data.DataSet	System.Data.DataSet
9	System.Data.DataTable	System.Data.DataTable
10	System.Data.DataRow	System.Data.DataRow
11	System.Data.DataColumn	System.Data.DataColumn

Connection Strings in ADO.NET

Connection string for SQL Server

- "data source=*Servernamehere*; integrated security=yes; initial catalog=*Databasenamehere*"

(or)
- "data source=*Servernamehere*; user id=*Usernamehere*; password=*Passwordhere*; initial catalog=*Databasenamehere*"

Connection string for Oracle

- "provider=msdaora.1; user id=*Usernamehere*; password=*Passwordhere*"

Connection string for MS Access

- @"provider=Microsoft.Ace.Oledb.12.0; data source=*FilePathHere*"

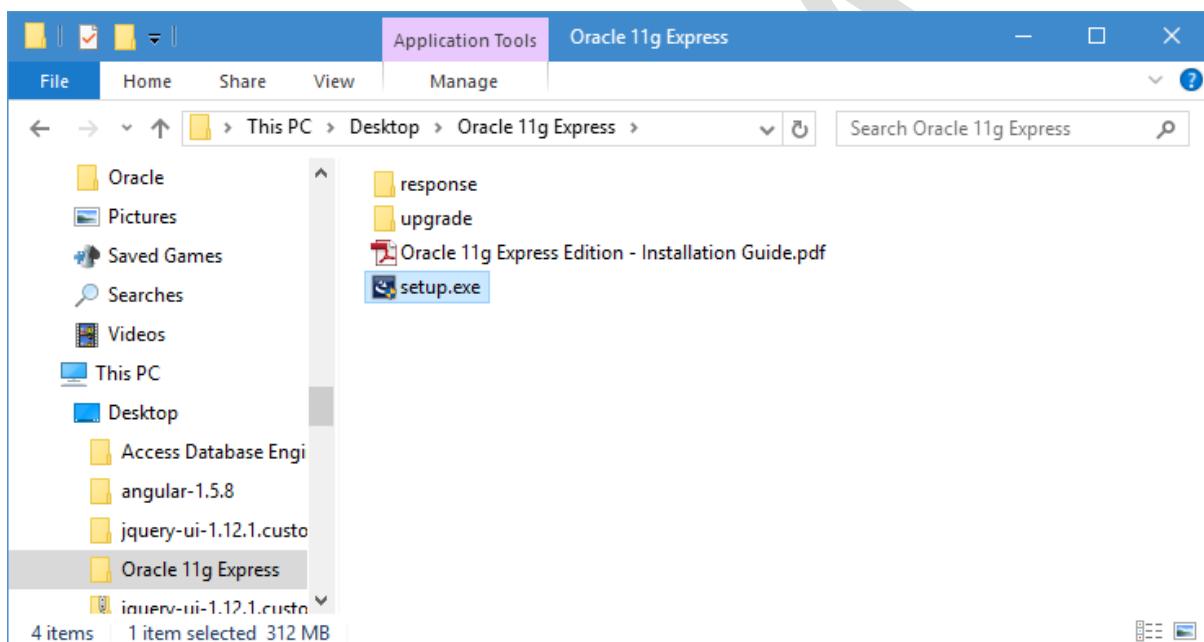
Connection string for MS Excel

- @"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=*FilePathHere*; Extended Properties='Excel 12.0;HDR=Yes;IMEX=1' "

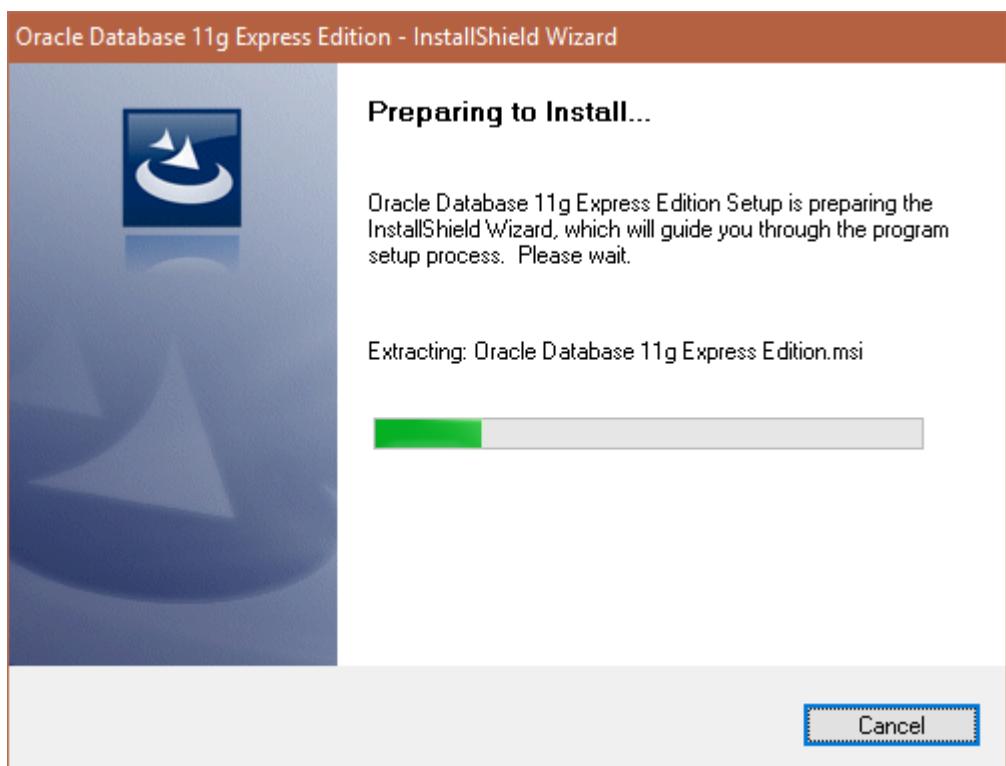
ADO.NET – Oracle - Example

Installing Oracle Database 11g Expression Edition

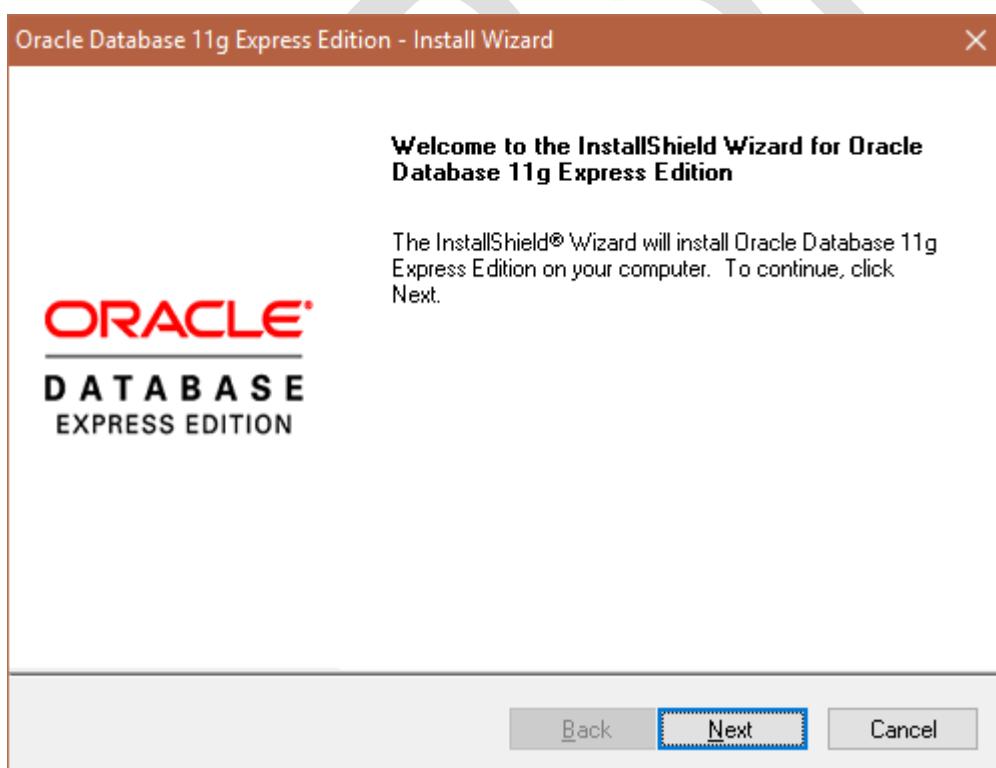
- Note: Ignore this step, if you have installed “Oracle 11g Express” already.
- You can download Oracle 11g Express Edition at:
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>
- Open “Oracle 11g Express” folder.



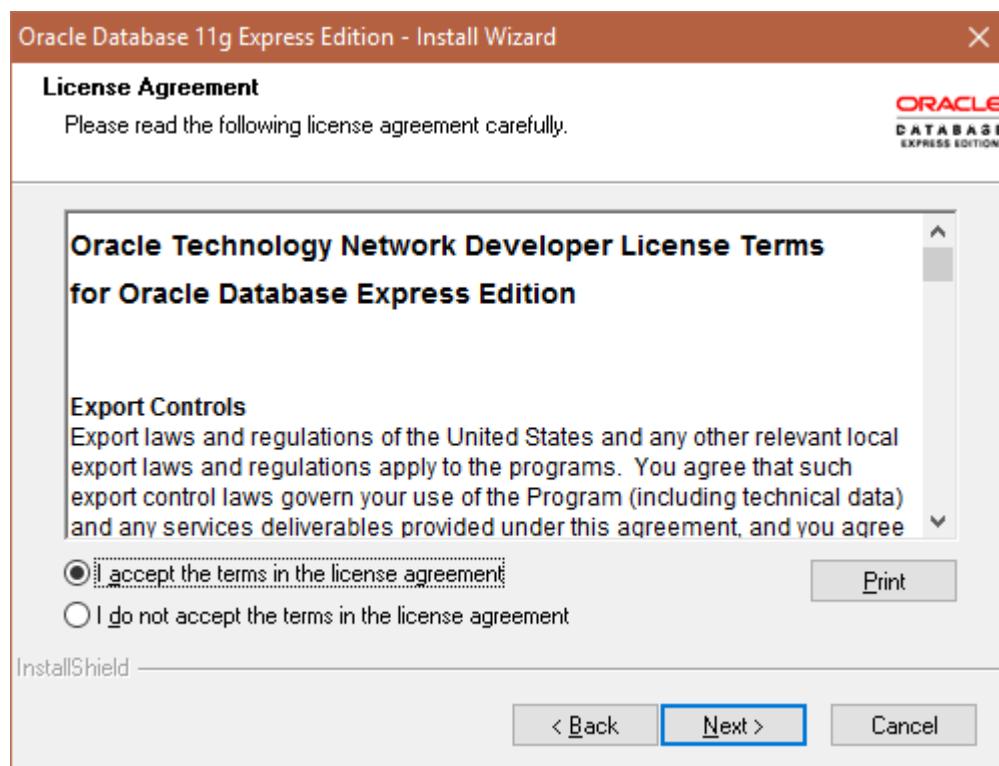
- Double click on “setup.exe”.
- Click on “Yes”.



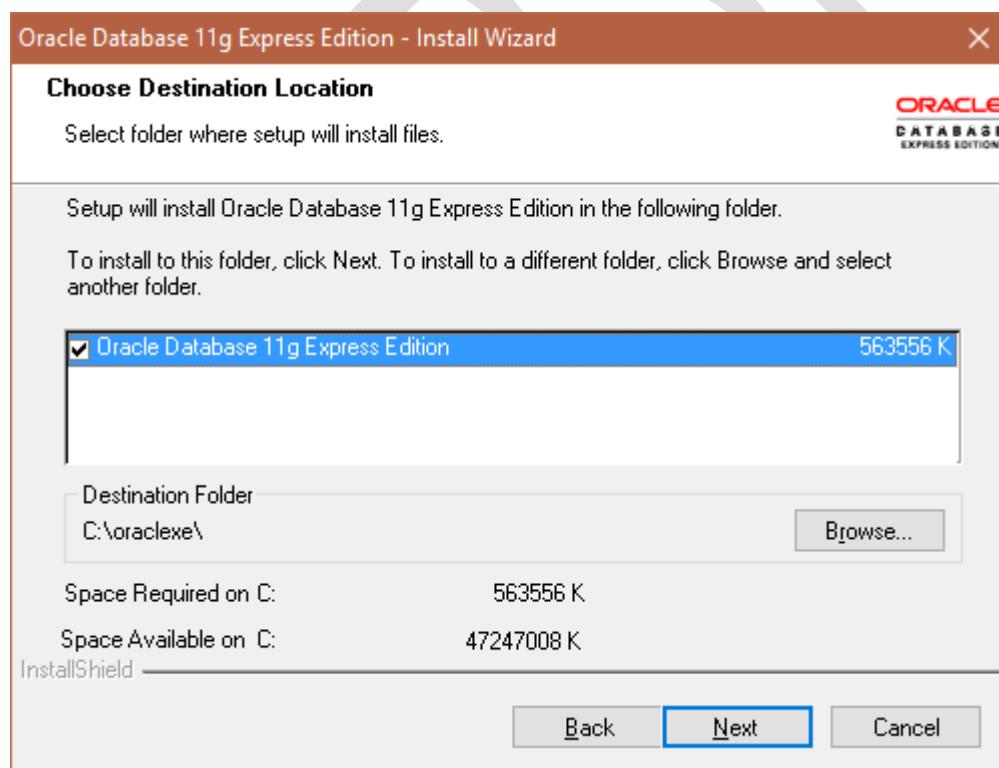
- Please wait while it loads...



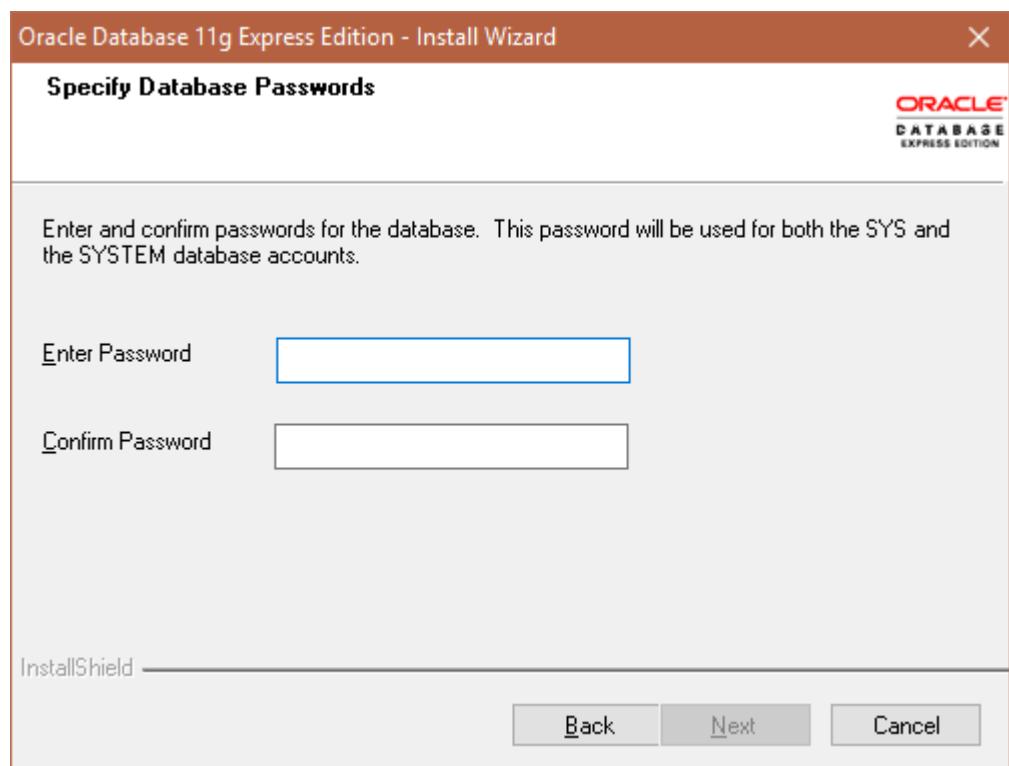
- Click on "Next".
- Click on "I accept the terms in the license agreement".



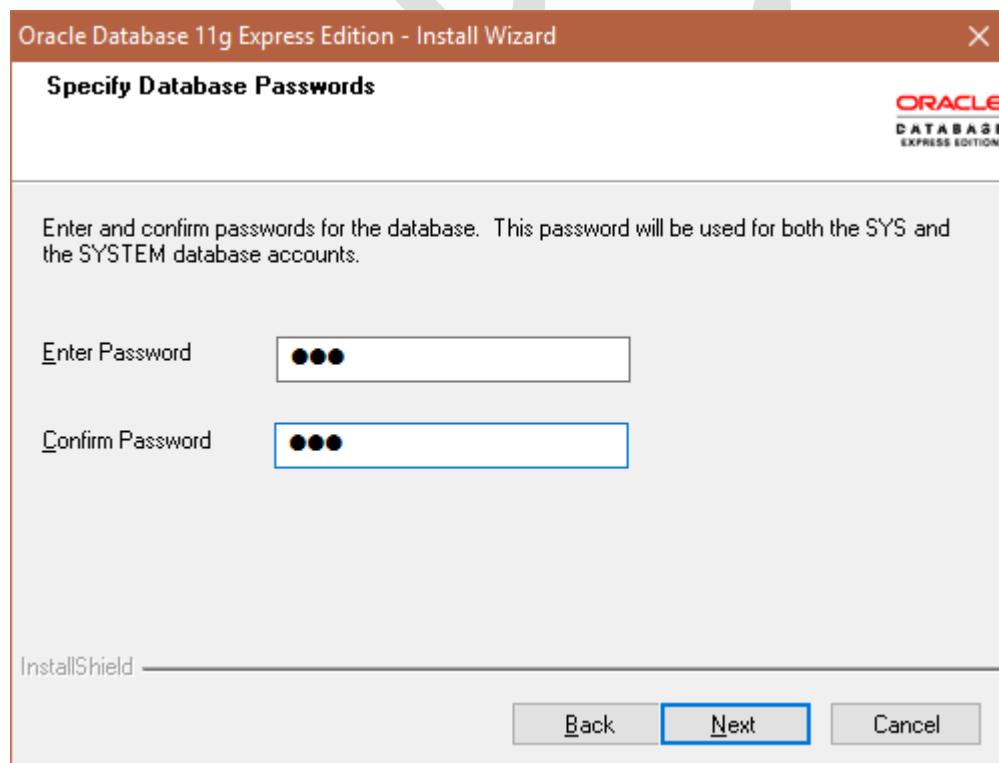
- Click on "Next".



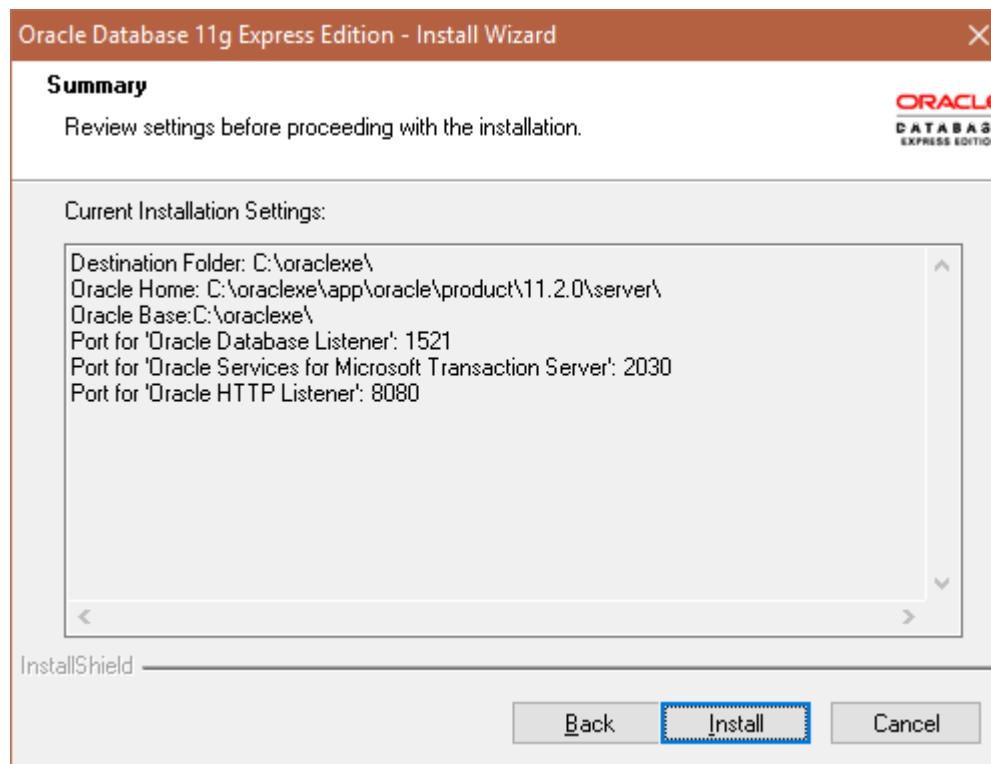
- Click on "Next".



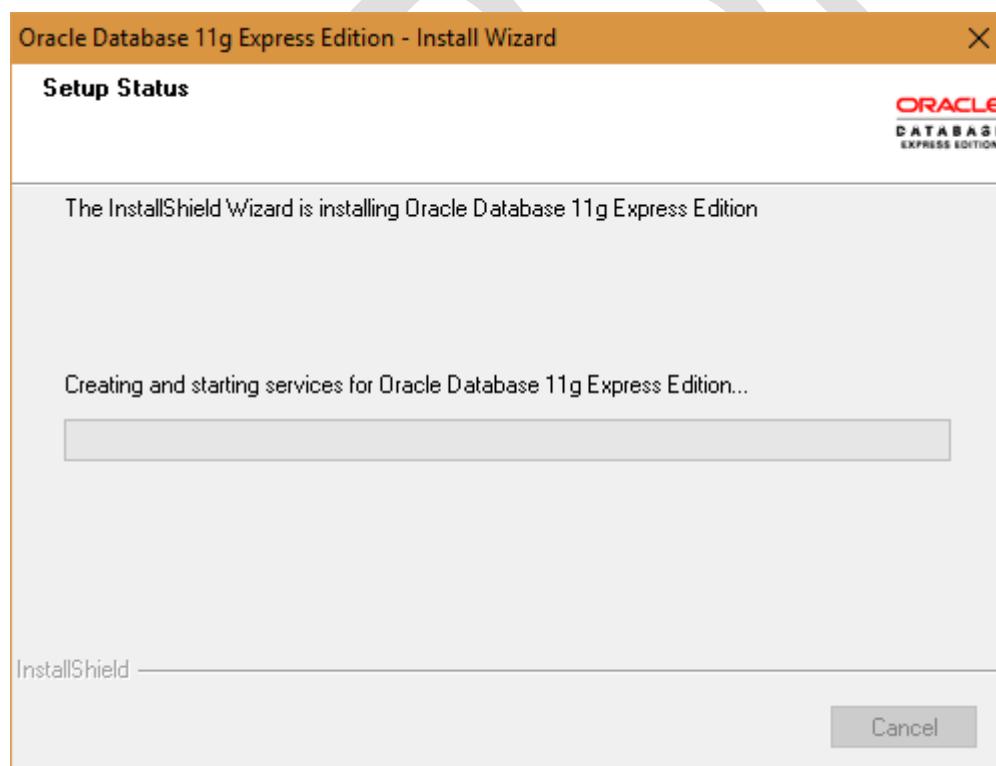
- Enter the password as "123".
- Enter the confirm password "123".



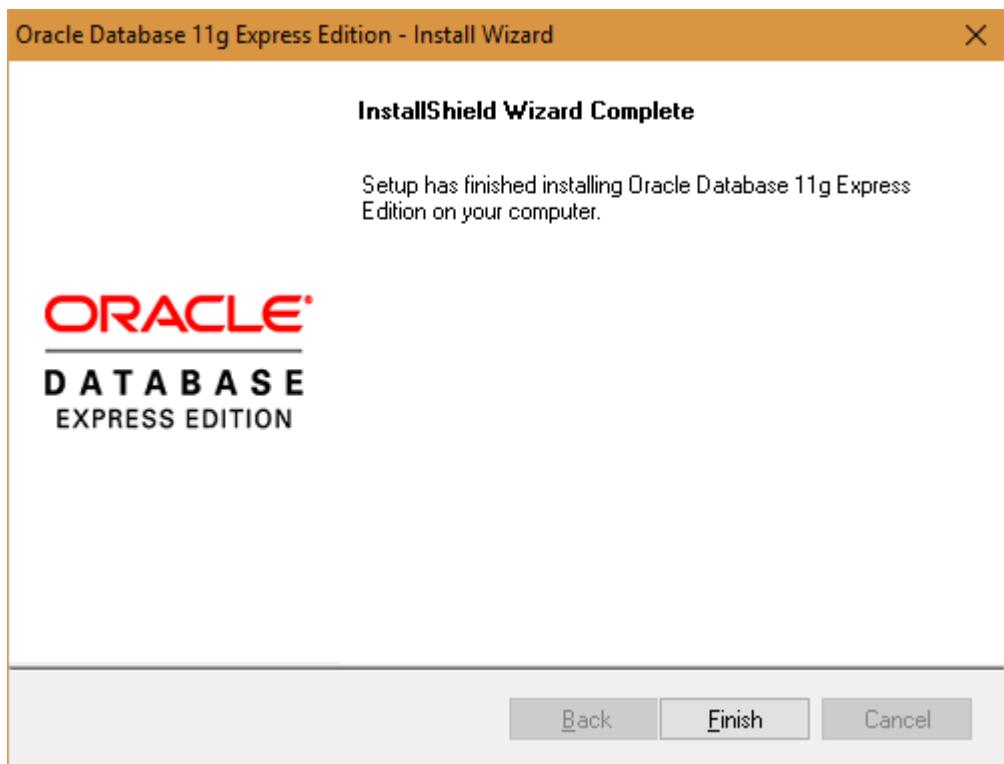
- Click on "Next".



- Click on "Install".



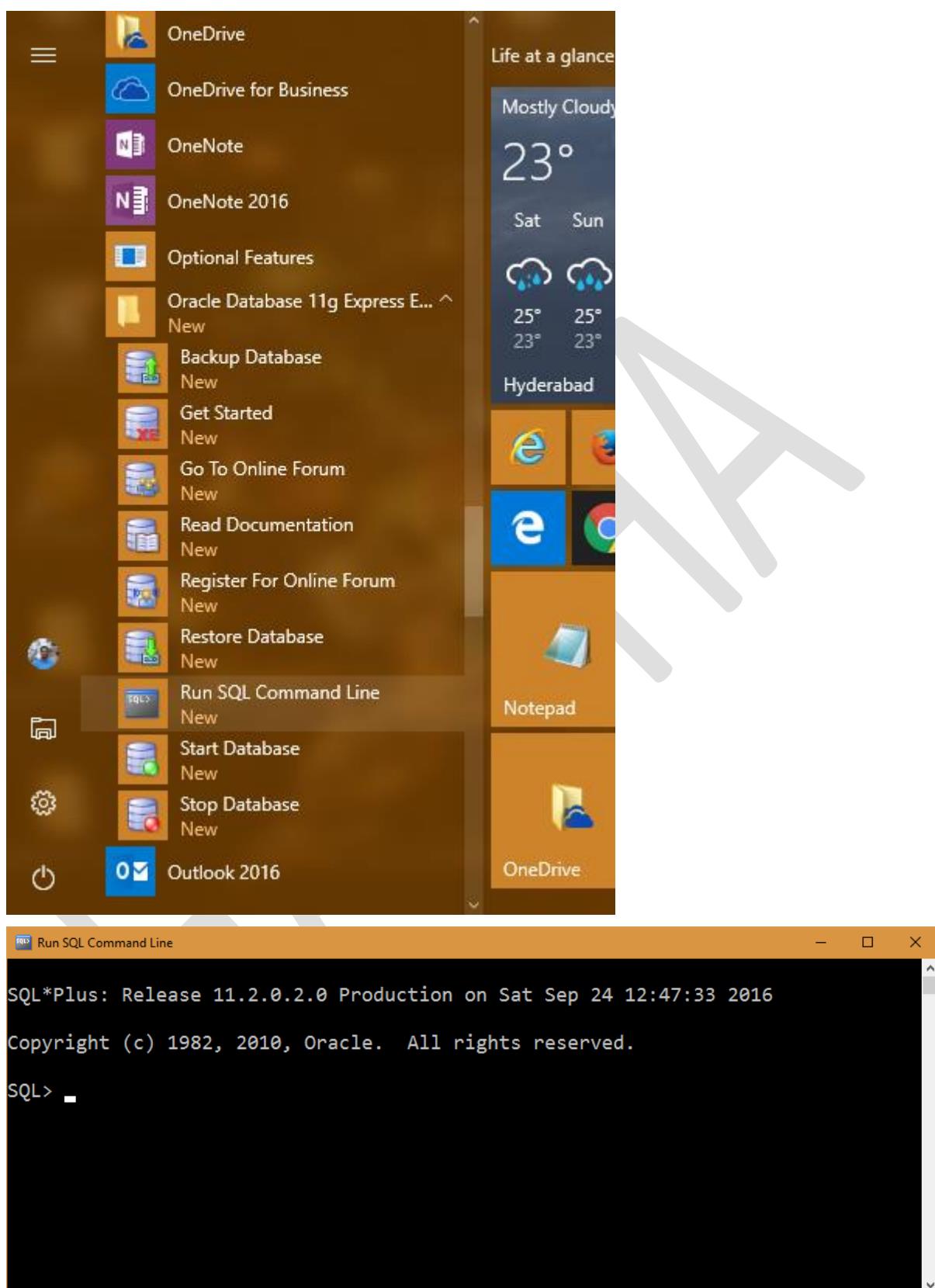
- Installation may take around 10 minutes.
 - Click on OK if it shows one or two errors while installing.



- Click on “Finish”.
- Restart the PC.

Creating table in Oracle

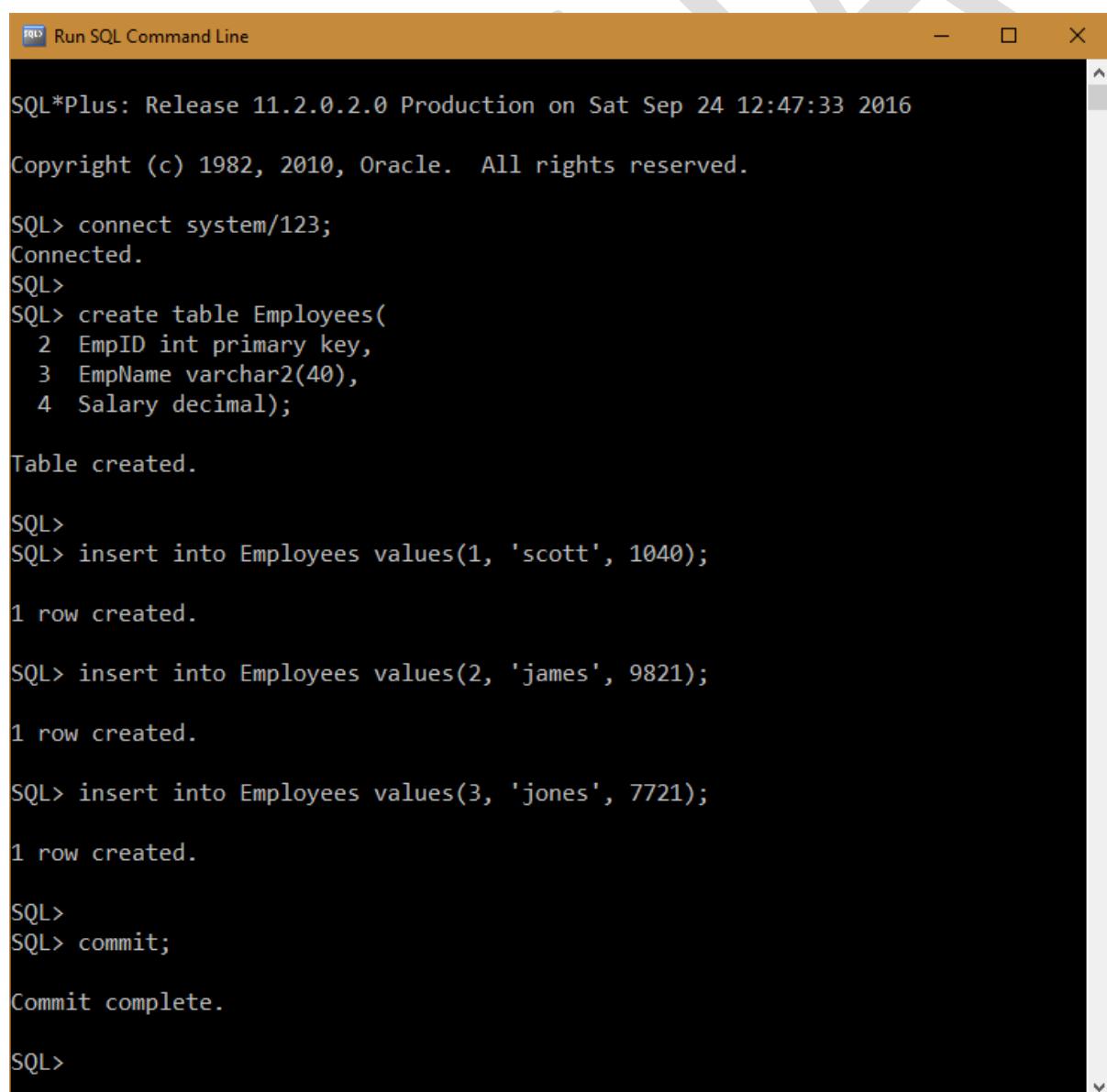
- Note: Ignore this step, if you have created “employees” table in Oracle already.
- Go to “Start” – “Oracle 11g Express Edition” – “Run SQL Command Line”.



- Type the following script and press Enter.

```
connect system/123;
```

```
create table Employees(  
    EmpID int primary key,  
    EmpName varchar2(40),  
    Salary decimal);  
  
insert into Employees values(1, 'scott', 1040);  
insert into Employees values(2, 'james', 9821);  
insert into Employees values(3, 'jones', 7721);  
  
commit;
```



The screenshot shows a terminal window titled "Run SQL Command Line" with the following content:

```
SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 24 12:47:33 2016  
Copyright (c) 1982, 2010, Oracle. All rights reserved.  
  
SQL> connect system/123;  
Connected.  
SQL>  
SQL> create table Employees(  
  2  EmpID int primary key,  
  3  EmpName varchar2(40),  
  4  Salary decimal);  
  
Table created.  
  
SQL>  
SQL> insert into Employees values(1, 'scott', 1040);  
1 row created.  
  
SQL> insert into Employees values(2, 'james', 9821);  
1 row created.  
  
SQL> insert into Employees values(3, 'jones', 7721);  
1 row created.  
  
SQL>  
SQL> commit;  
Commit complete.  
  
SQL>
```

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “OracleExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OracleExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.OleDb;
using System.Data;

namespace OracleExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            OleDbConnection cn;
            OleDbCommand cmd;
            OleDbDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            /* create objects */
            cn = new OleDbConnection();
```

```
cmd = new OleDbCommand();
adp = new OleDbDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "user id=system; password=123;
provider=msdaora.1";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

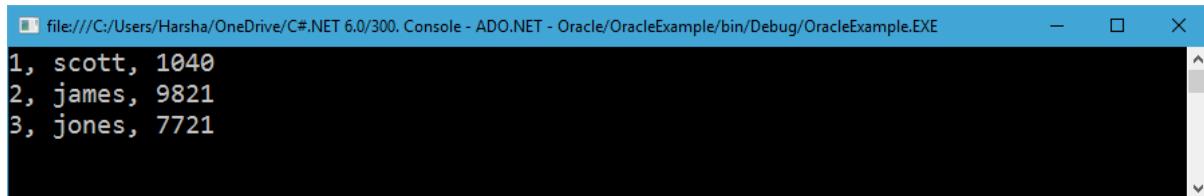
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    Console.WriteLine(eid + "," + ename + "," + sal);
}
Console.ReadKey();
}

}
```

Running the Project

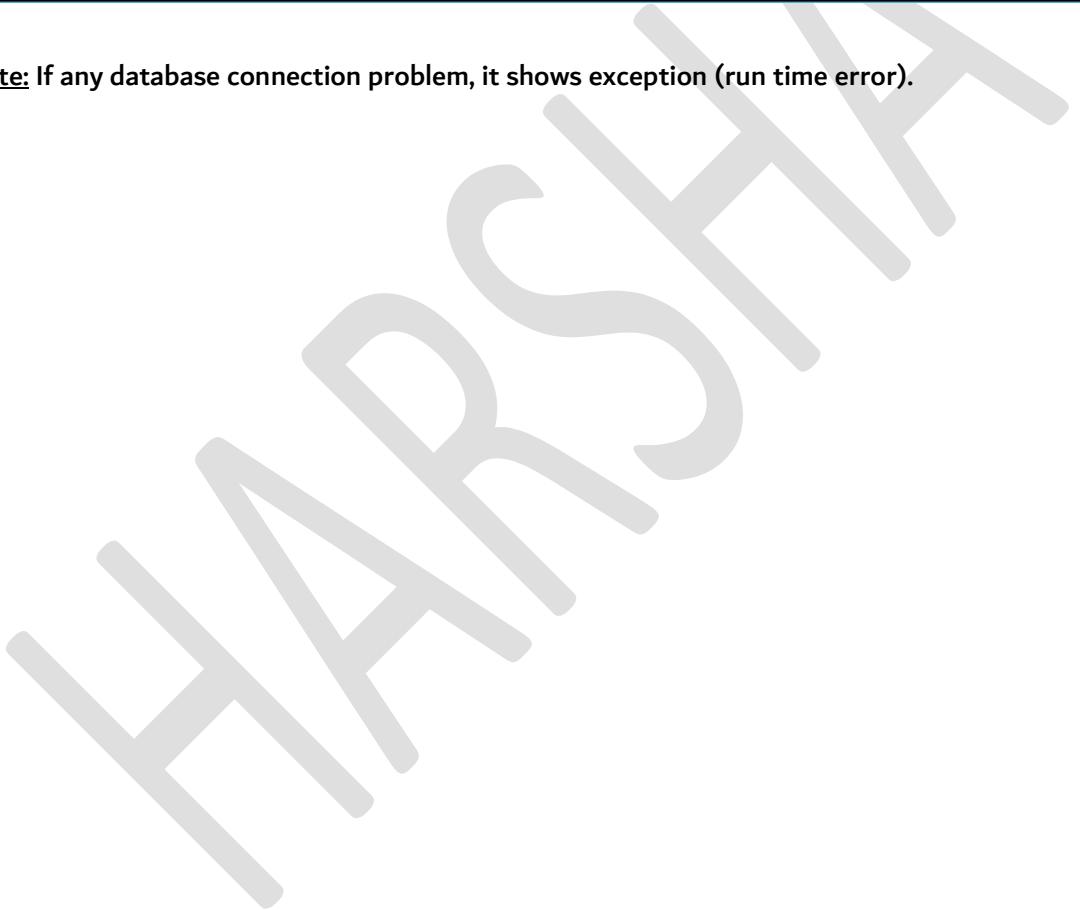
- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/300. Console - ADO.NET - Oracle/OracleExample/bin/Debug/OracleExample.EXE
1, scott, 1040
2, james, 9821
3, jones, 7721
```

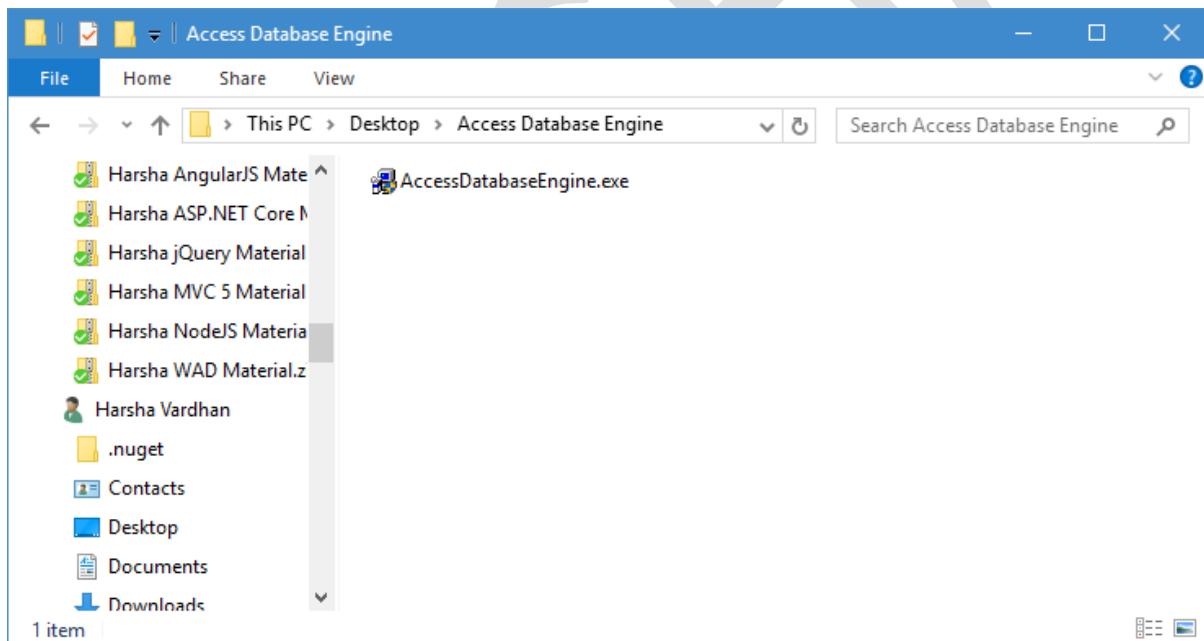
Note: If any database connection problem, it shows exception (run time error).



ADO.NET – MS Access - Example

Installing Access Database Engine

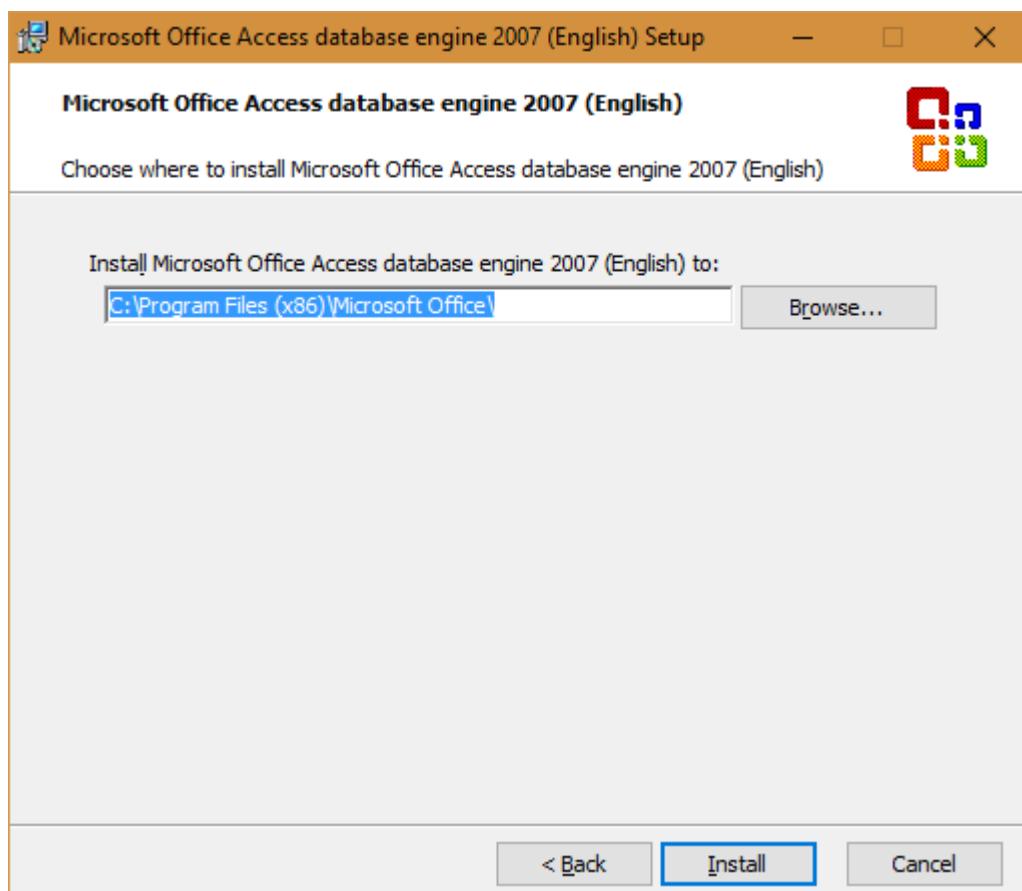
- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at: <https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”.
- Note: Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>
- To install “Access Database Engine”, go to “Access Database Engine” folder.



- Double click on “AccessDatabaseEngine”.
- Click on “Yes”.



- Check the checkbox "I accept the terms in the License Agreement".
- Click on "Next".



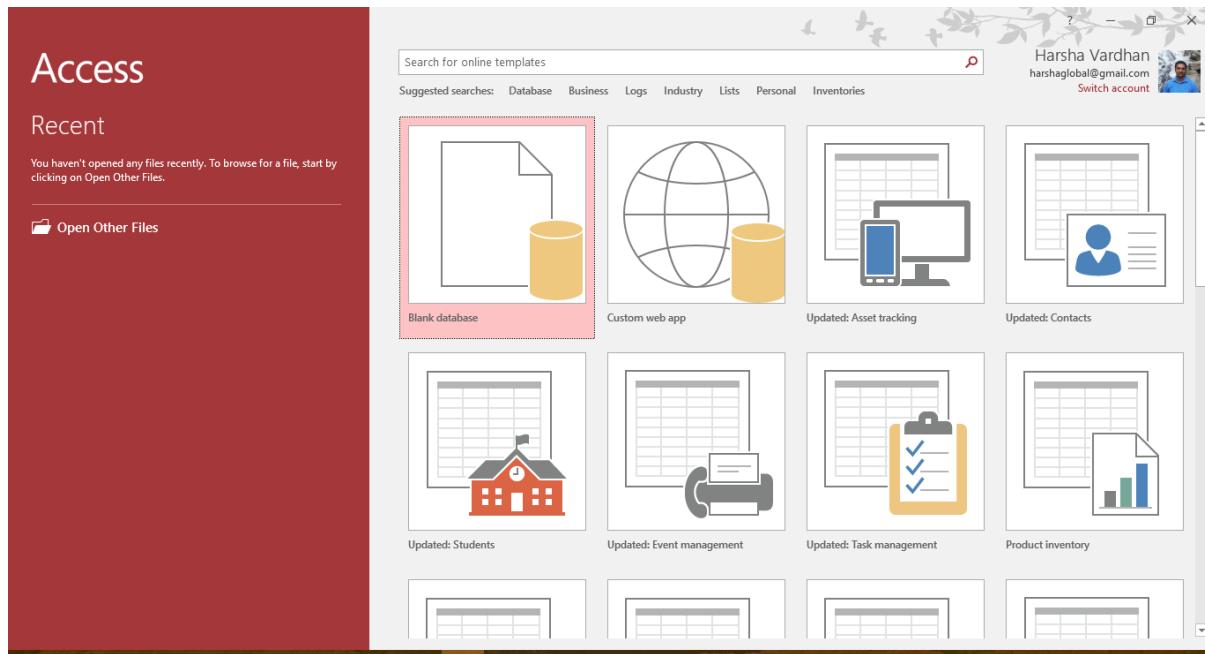
- Click on "Install".
- Installation may take around 1 or 2 minutes.



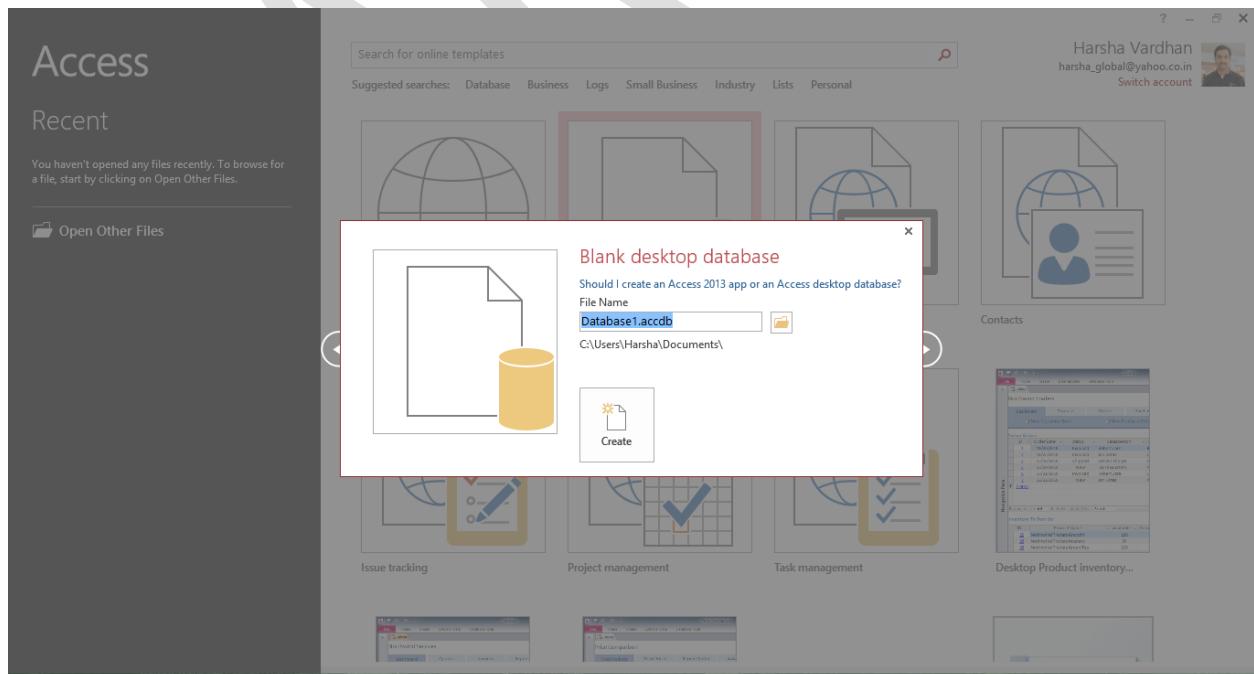
- Click on OK.

Creating table in MS Access

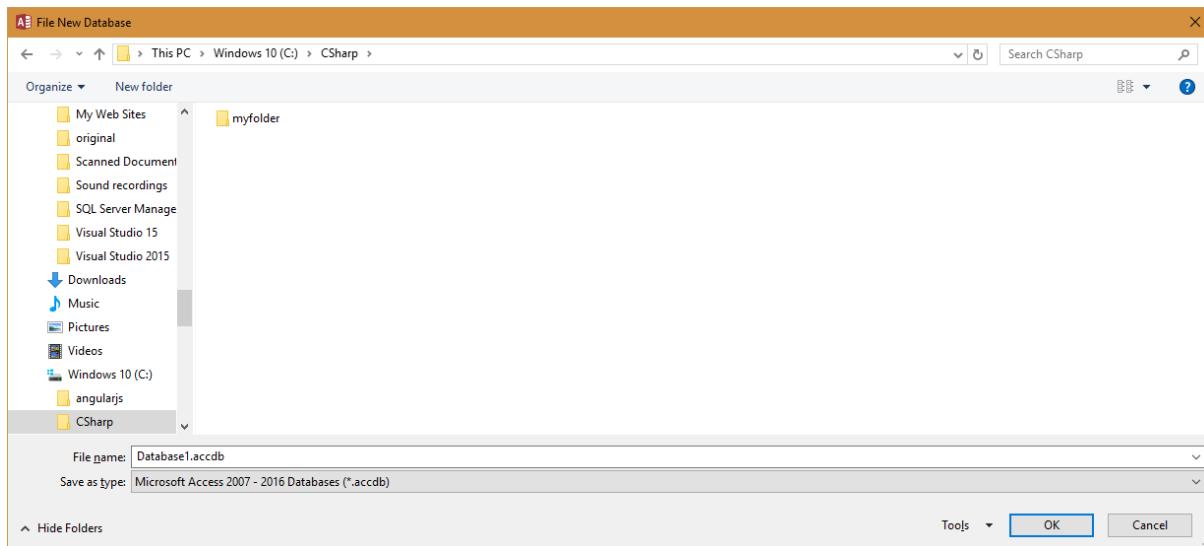
- Go to “Start” – “Access 2016”.



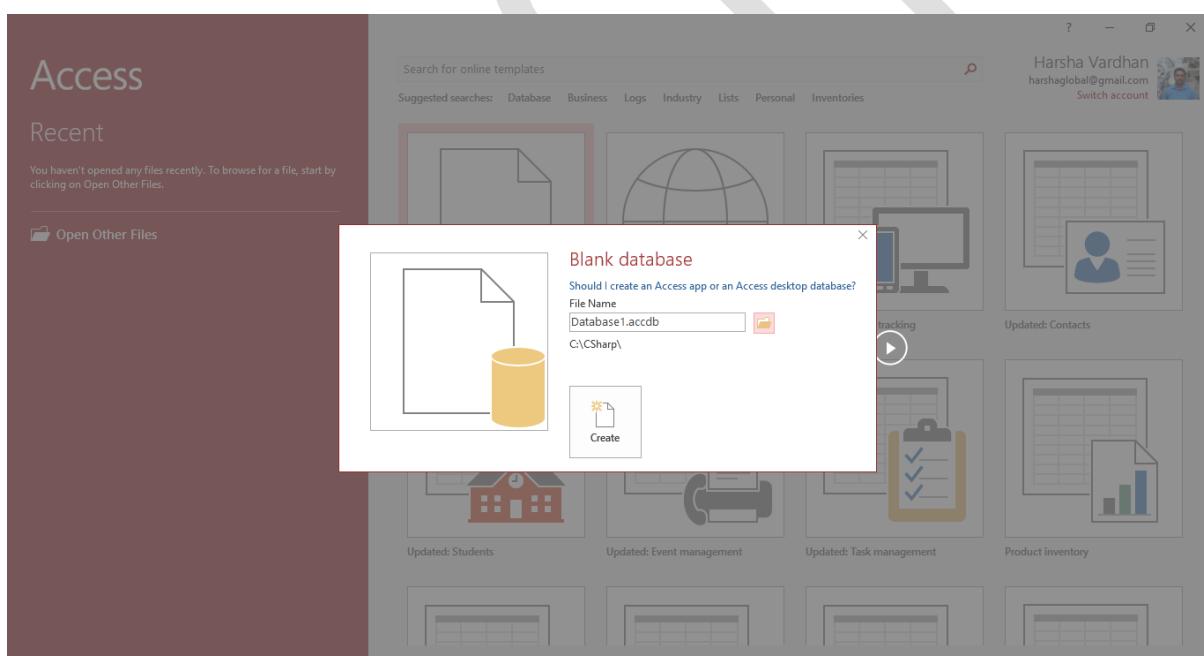
- Click on “Blank desktop database”.
- Type the file name as “Database1.accdb”.



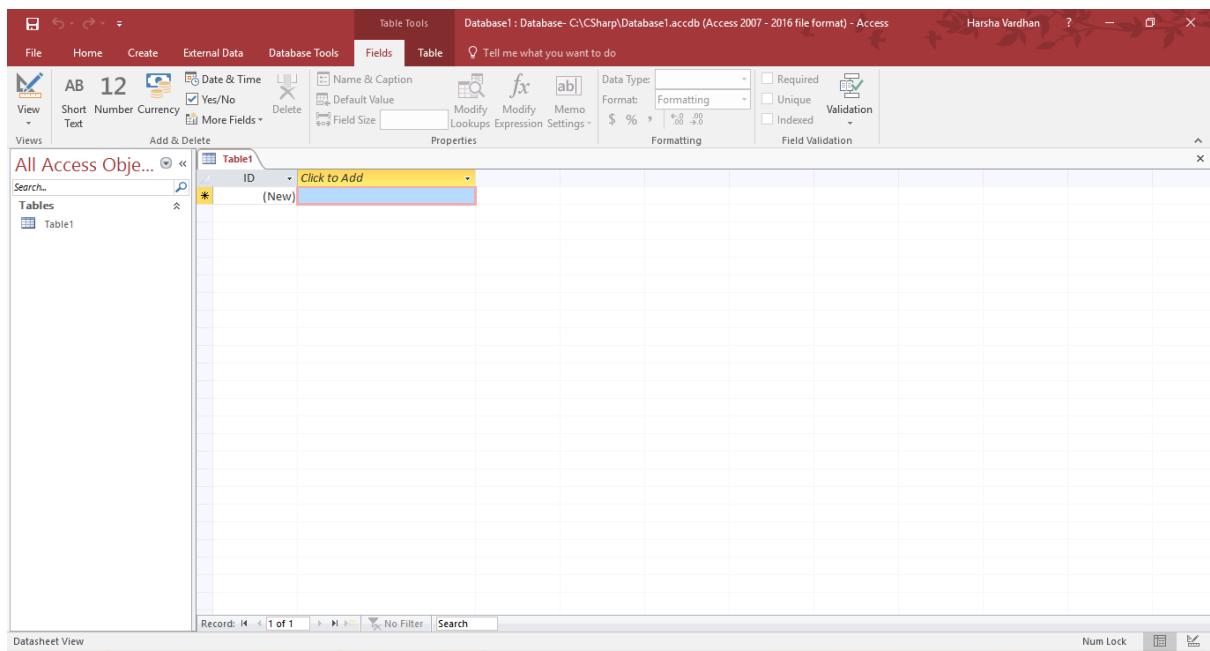
- Click on the folder icon and select “C:\CSharp” folder.



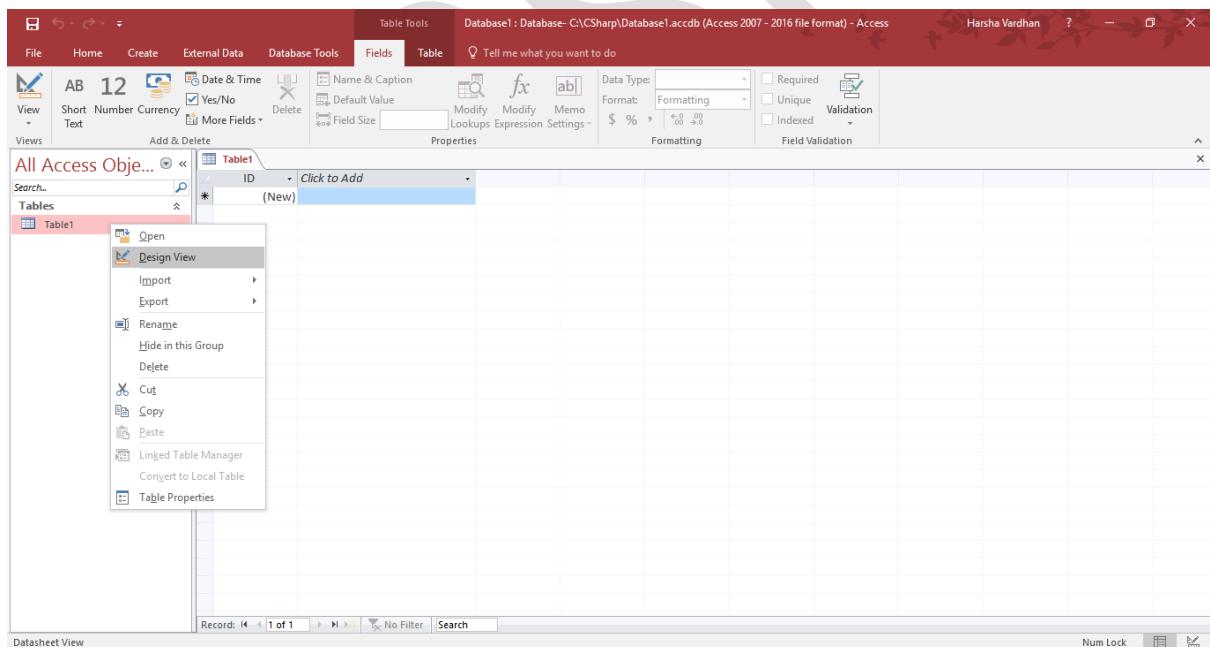
- Click on OK.



- Click on "Create".



- Right click on “Table1” and click on “Design View”.



- Type the table name as “Employees”.



- Click on OK.
- Type the table structure as follows:

A screenshot of Microsoft Access in Design view. The ribbon shows 'Database Tools' and 'Design'. The left pane shows 'All Access Objects' with 'Tables' expanded and 'Employees' selected. The main area displays the 'Employees' table structure:

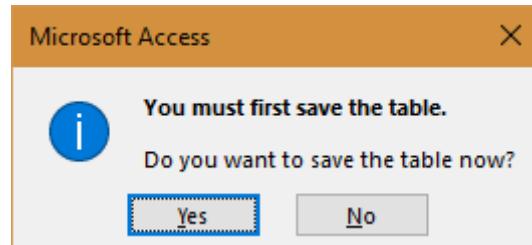
Field Name	Data Type	Description (Optional)
EmpID	AutoNumber	
EmpName	Short Text	
Salary	Number	

The 'Field Properties' pane at the bottom right contains the note: 'A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.'

- Right click "Employees" and click on "Open".

A screenshot of Microsoft Access showing a context menu for the 'Employees' table. The menu is open over the table name in the list pane. The 'Open' option is highlighted with a blue border. Other options include 'Design View', 'Import', 'Export', 'Rename', 'Hide in this Group', 'Delete', 'Cut', 'Copy', 'Paste', 'Linked Table Manager', and 'Table Properties'.

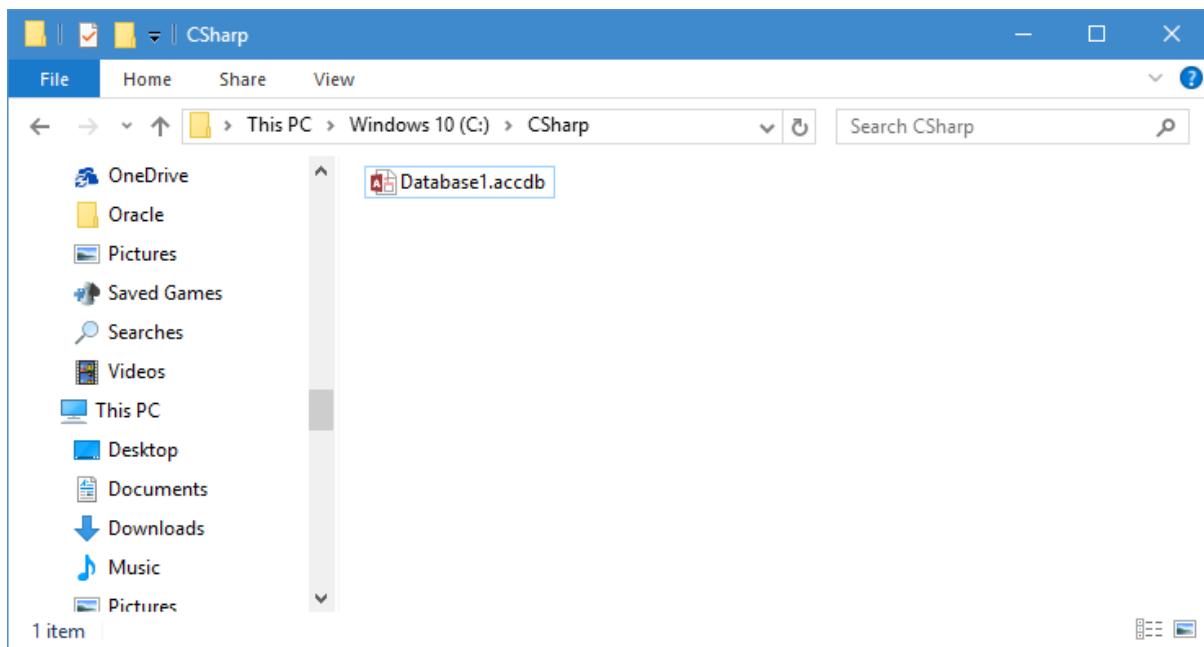
- Click on “Yes”.



- Type the data as follows:

A screenshot of Microsoft Access 2016 showing the "Employees" table. The table has four columns: EmpID, EmpName, Salary, and Click to Add. There are three data rows: 1 Scott, 2 Allen, and 3 Jones. A new record row is visible at the bottom with values {New}, 0, and Click to Add. The "Salary" column is highlighted with a yellow background.

- Save the file.
- Close “Microsoft Access 2016”.
- Make sure “Database1.accdb” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MSAccessExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MSAccessExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.OleDb;
using System.Data;

namespace MSAccessExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            OleDbConnection cn;
            OleDbCommand cmd;
            OleDbDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            /* create objects */
            cn = new OleDbConnection();
```

```
cmd = new OleDbCommand();
adp = new OleDbDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = @"provider=Microsoft.Ace.Oledb.12.0; data
source=C:\CSharp\Database1.accdb";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

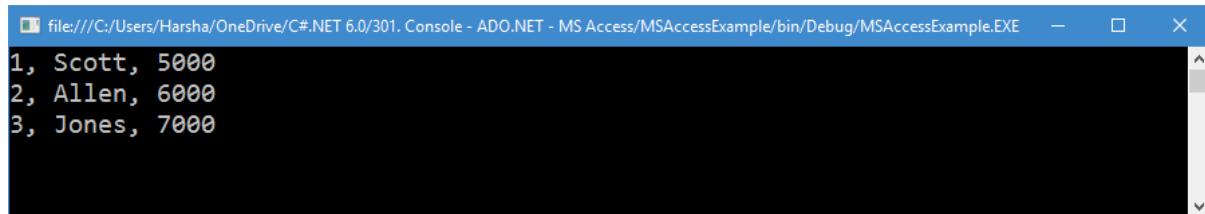
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    Console.WriteLine(eid + "," + ename + "," + sal);
}
Console.ReadKey();
}

}
```

Running the Project

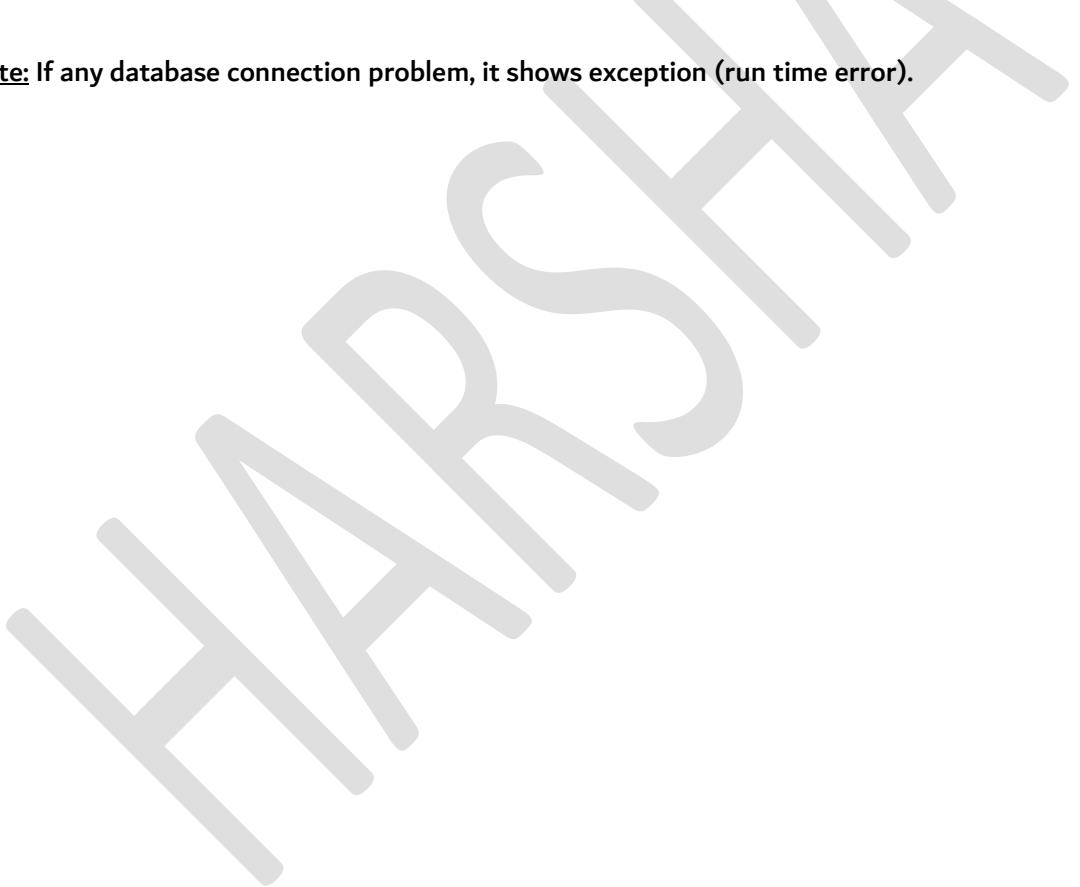
- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1, Scott, 5000
2, Allen, 6000
3, Jones, 7000
```

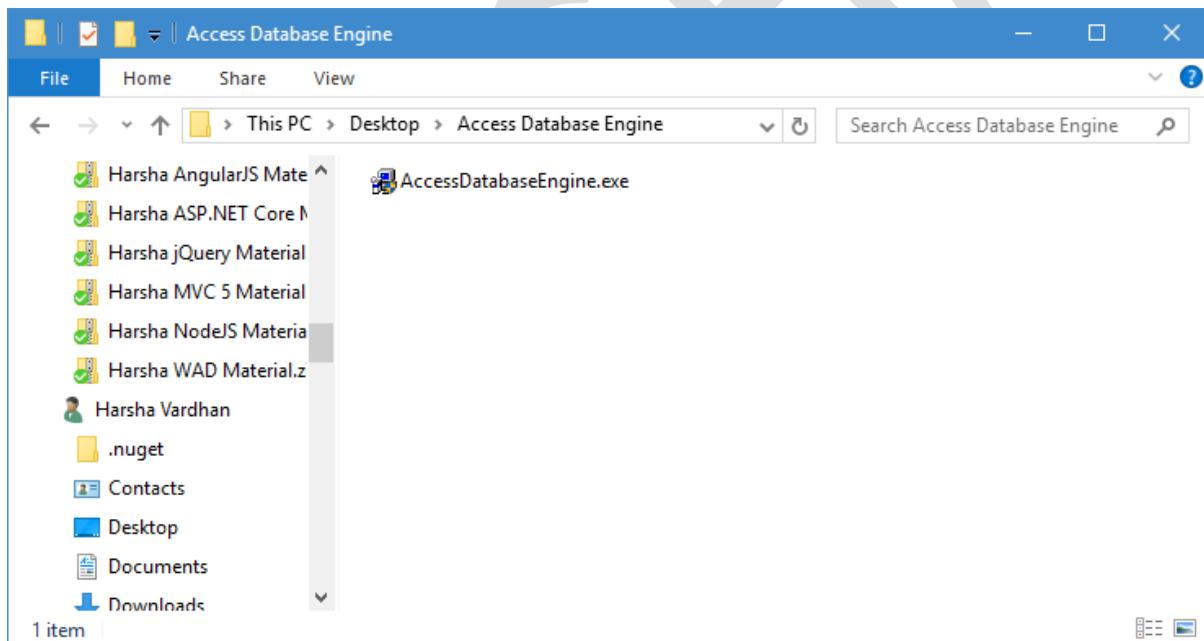
Note: If any database connection problem, it shows exception (run time error).



ADO.NET – MS Excel - Example

Installing Access Database Engine

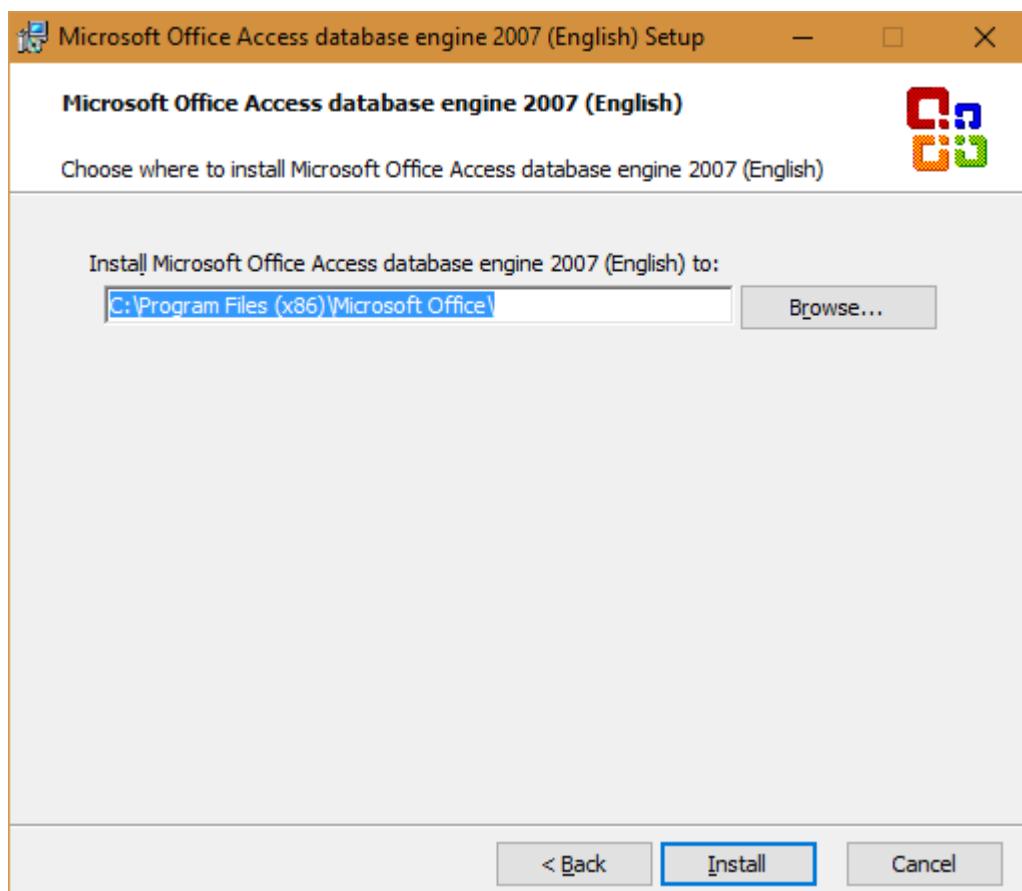
- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at:
<https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”.
- Note: Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>
- To install “Access Database Engine”, go to “Access Database Engine” folder.



- Double click on “AccessDatabaseEngine”.
- Click on “Yes”.



- Check the checkbox “I accept the terms in the License Agreement”.
- Click on “Next”.



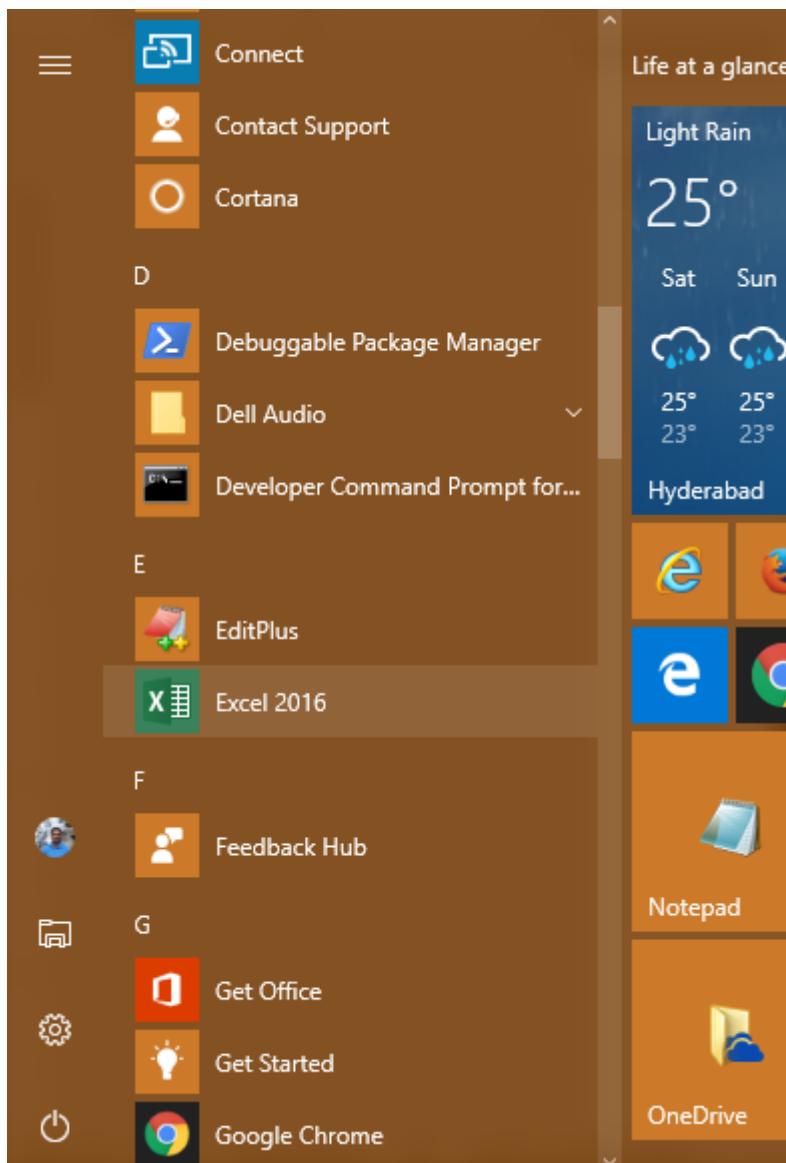
- Click on "Install".
- Installation may take around 1 or 2 minutes.



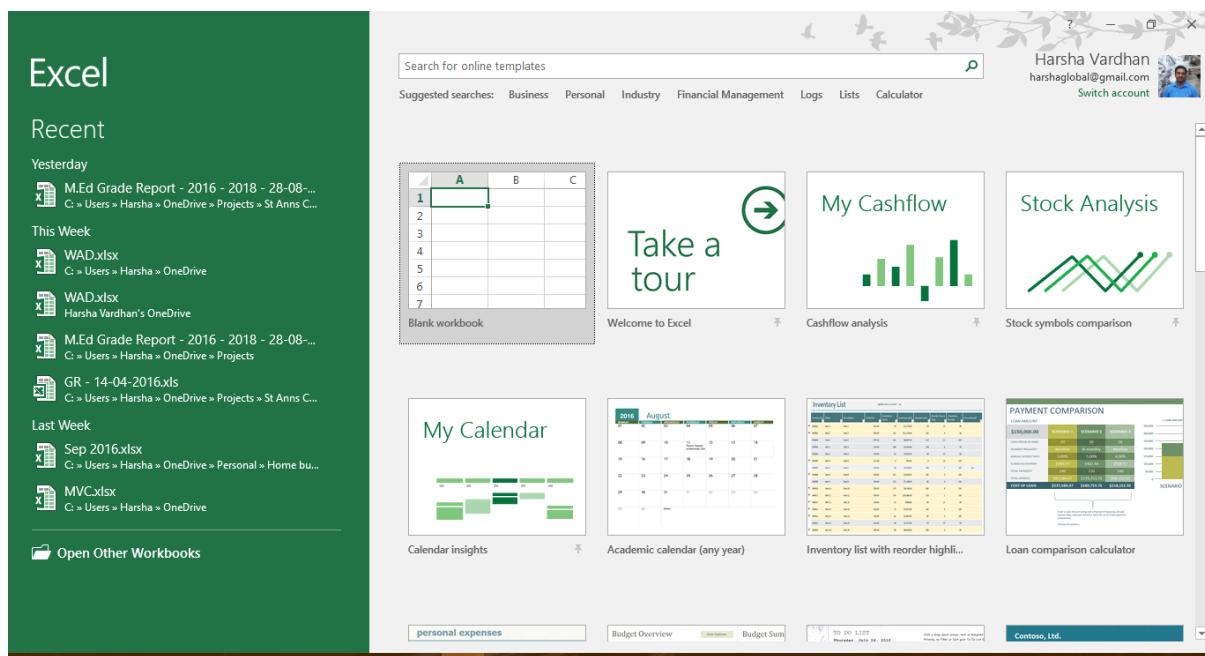
- Click on OK.

Creating work book in MS Excel

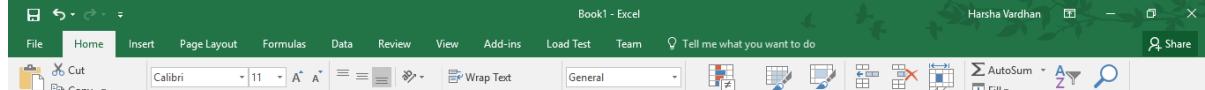
- Go to “Start” – “Excel 2016”.



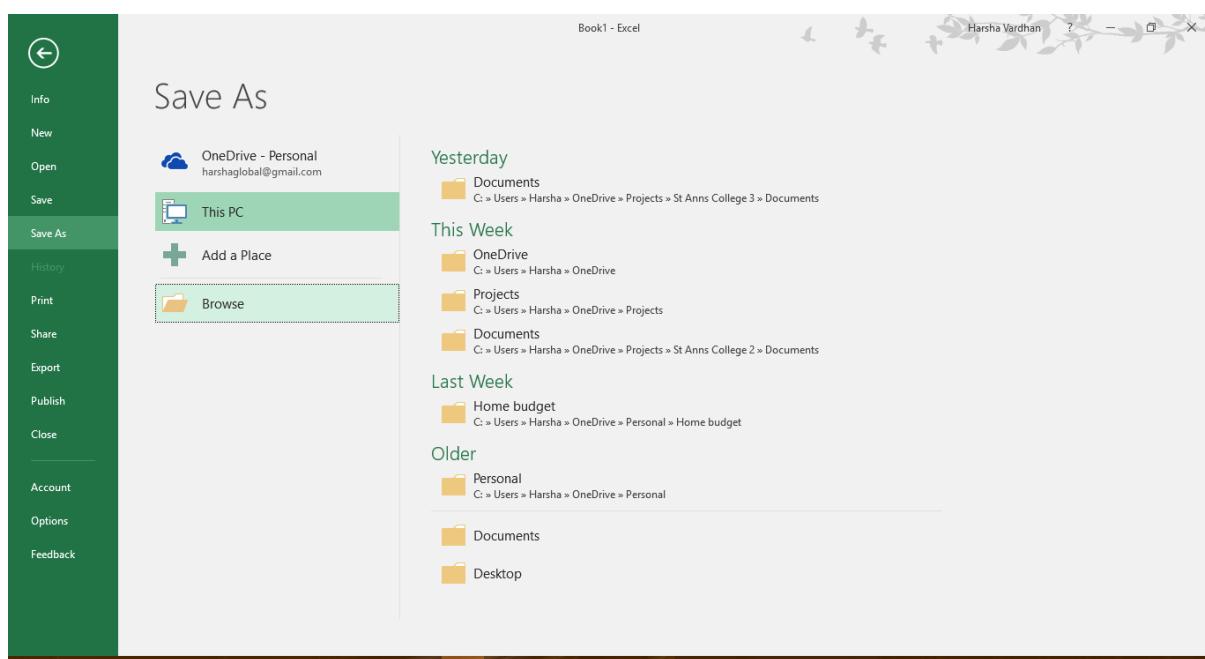
- Click on “Blank Workbook”.



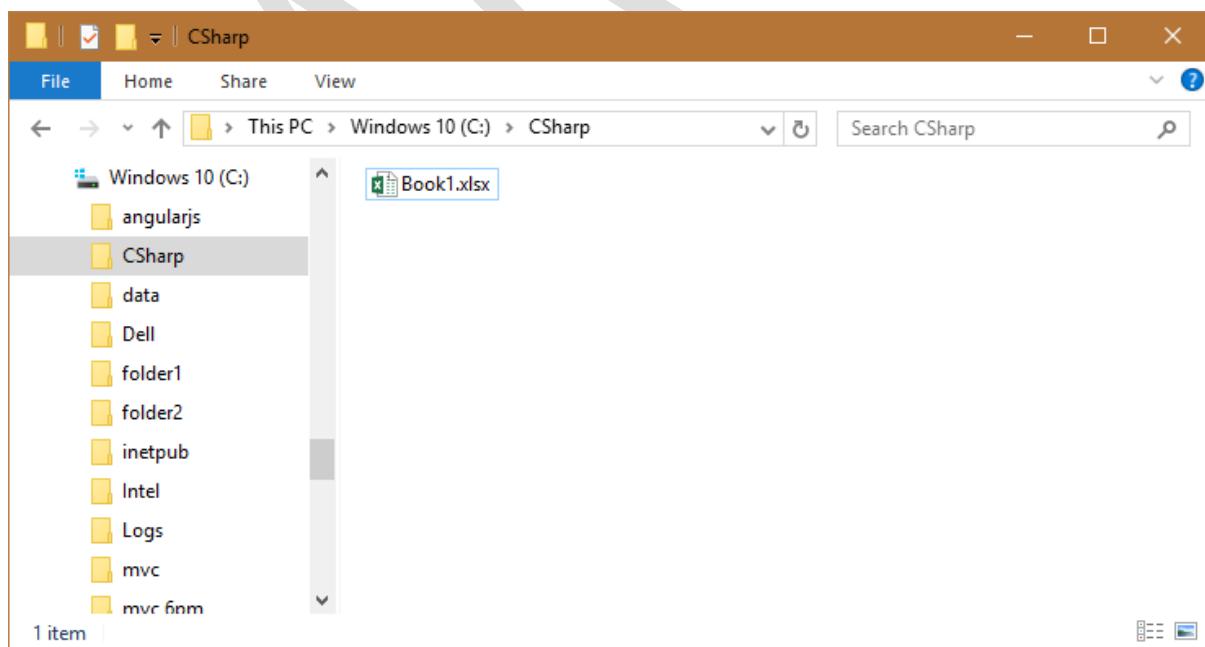
- Type the data as follows:

Book1 - Excel											
											
1	EmplID	EmpName	Salary								
2	1	abc	1000								
3	2	def	2000								
4	3	ghi	3000								
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
Sheet1											

- Go to "File" – "Save" – "This PC" – "Browse".



- Select the folder as “C:\CSharp”.
- Type the filename as “Book1.xlsx”.
- Click on “Save”.
- Close “Microsoft Excel 2016”.
- Make sure “Book1.xlsx” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “MSEExcelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MSEExcelExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.OleDb;
using System.Data;

namespace MSEExcelExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            OleDbConnection cn;
            OleDbCommand cmd;
            OleDbDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            /* create objects */
            cn = new OleDbConnection();
```

```
cmd = new OleDbCommand();
adp = new OleDbDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\CSharp\Book1.xlsx; Extended Properties='Excel
12.0;HDR=Yes;IMEX=1'";
cmd.CommandText = "select * from [Sheet1$]";
cmd.Connection = cn;
adp.SelectCommand = cmd;

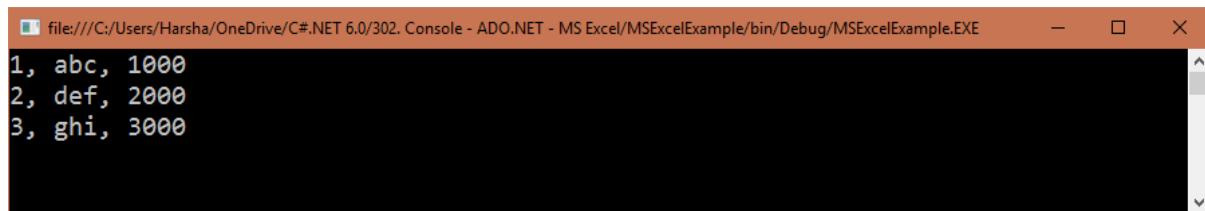
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    Console.WriteLine(eid + ", " + ename + ", " + sal);
}
Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1, abc, 1000
2, def, 2000
3, ghi, 3000
```

Note: If any database connection problem, it shows exception (run time error).

Large watermark text "HARSH" is visible diagonally across the page.

ADO.NET – SQL Server to Oracle - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

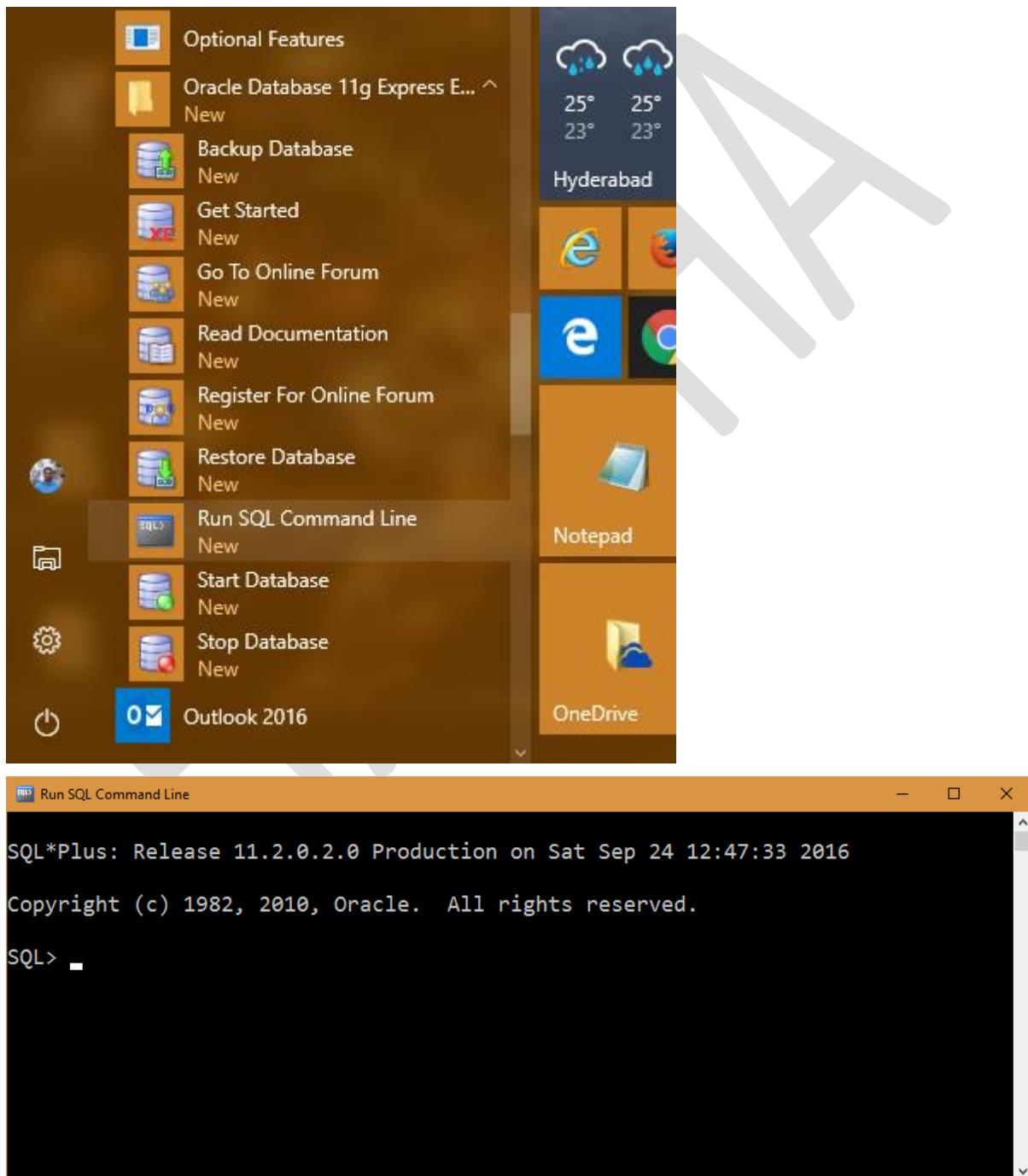
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Installing Oracle Database 11g Expression Edition

- Install “Oracle 11g Express” as shown in the previous example.
- You can download Oracle 11g Express Edition at:
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

Creating table in Oracle

- Note: Ignore this step, if you have created “employees2” table in Oracle already.
- Go to “Start” – “Oracle 11g Express Edition” – “Run SQL Command Line”.

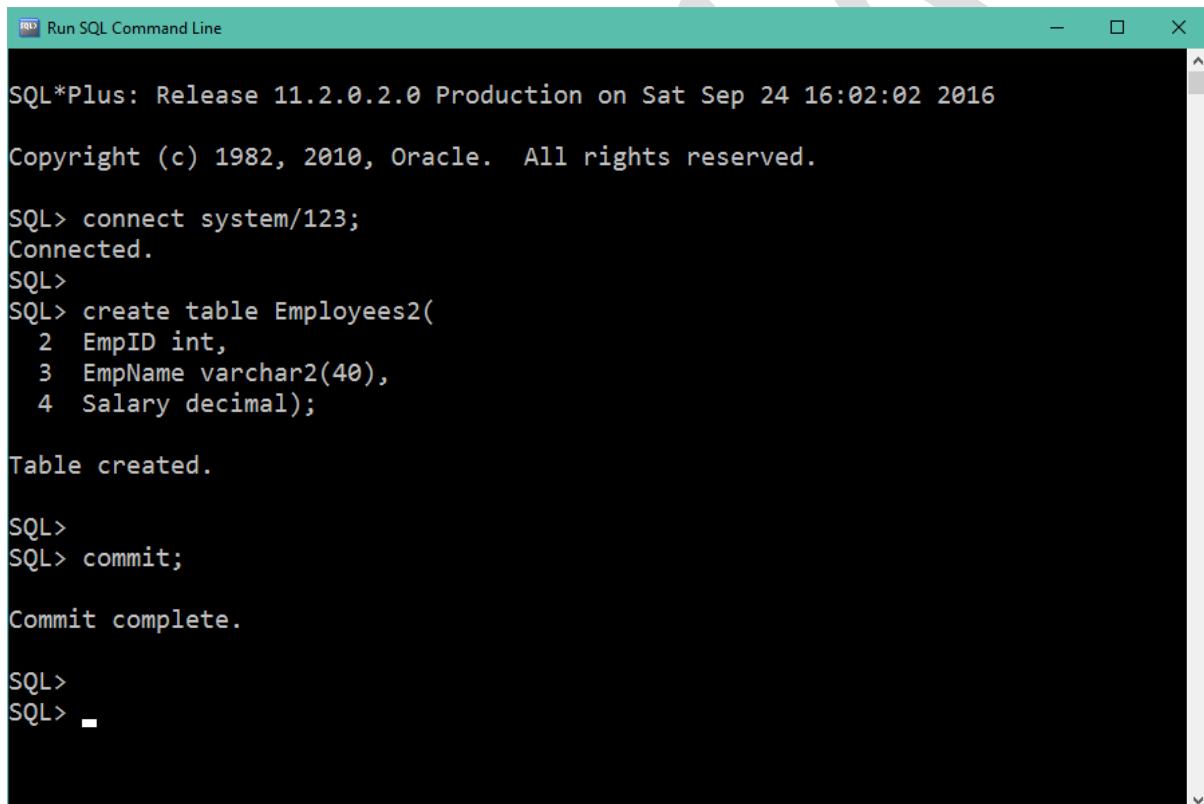


- Type the following script and press Enter.

```
connect system/123;
```

```
create table Employees2(
    EmpID int,
    EmpName varchar2(40),
    Salary decimal);
```

```
commit;
```



The screenshot shows a terminal window titled "Run SQL Command Line" displaying SQL*Plus output. The session starts with the Oracle release information: "SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 24 16:02:02 2016". It then shows the copyright notice: "Copyright (c) 1982, 2010, Oracle. All rights reserved." The user connects to the "system" schema with password "123", which is successful ("Connected"). The user then creates a new table named "Employees2" with three columns: "EmpID" (integer), "EmpName" (variable character string of length 40), and "Salary" (decimal). After the creation command, the message "Table created." is displayed. Finally, the user commits the transaction with the command "commit;". The session ends with the prompt "SQL>" followed by a blank line.

```
SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 24 16:02:02 2016
Copyright (c) 1982, 2010, Oracle. All rights reserved.

SQL> connect system/123;
Connected.
SQL>
SQL> create table Employees2(
 2   EmpID int,
 3   EmpName varchar2(40),
 4   Salary decimal);

Table created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> -
```

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SqlServerToOracleExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerToOracleExample”.
- Click on OK.

Program.cs

```
//Copy data from "SqlServer" to "Oracle"
using System;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;

namespace SqlServerToOracleExample
{
    class Program
    {
        static void Main()
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;
```

```

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);

    InsertIntoOracle(eid, ename, sal);
}

Console.WriteLine(dt.Rows.Count + " records copied");

```

```

        Console.ReadKey();
    }

private static void InsertIntoOracle(int eid, string ename, decimal sal)
{
    //create reference variables
    OleDbConnection cn;
    OleDbCommand cmd;

    //create objects
    cn = new OleDbConnection();
    cmd = new OleDbCommand();

    //calling properties
    cn.ConnectionString = "user id=system; password=123;
provider=msdaora.1";
    cmd.CommandText = string.Format("insert into Employees2
values({0},'{1}',{2})", eid, ename, sal);
    cmd.Connection = cn;

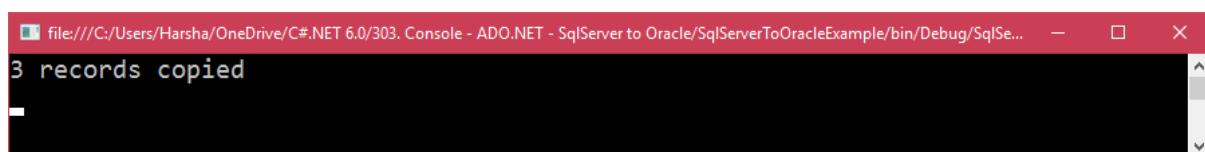
    //calling methods
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows Command Prompt window with the following text displayed:

```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/303. Console - ADO.NET - SqlServer to Oracle/SqlServerToOracleExample/bin/Debug/SqlSe...
3 records copied
-
```

Note: If any database connection problem, it shows exception (run time error).

Check the data in “Oracle”:



```
Run SQL Command Line
SQL> select * from Employees2;
  EMPID EMPNAME          SALARY
-----  -----
      3 Jones            6000
      1 Scott             4000
      2 Allen             5000
SQL>
```

ADO.NET – SQL Server to MS Excel - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

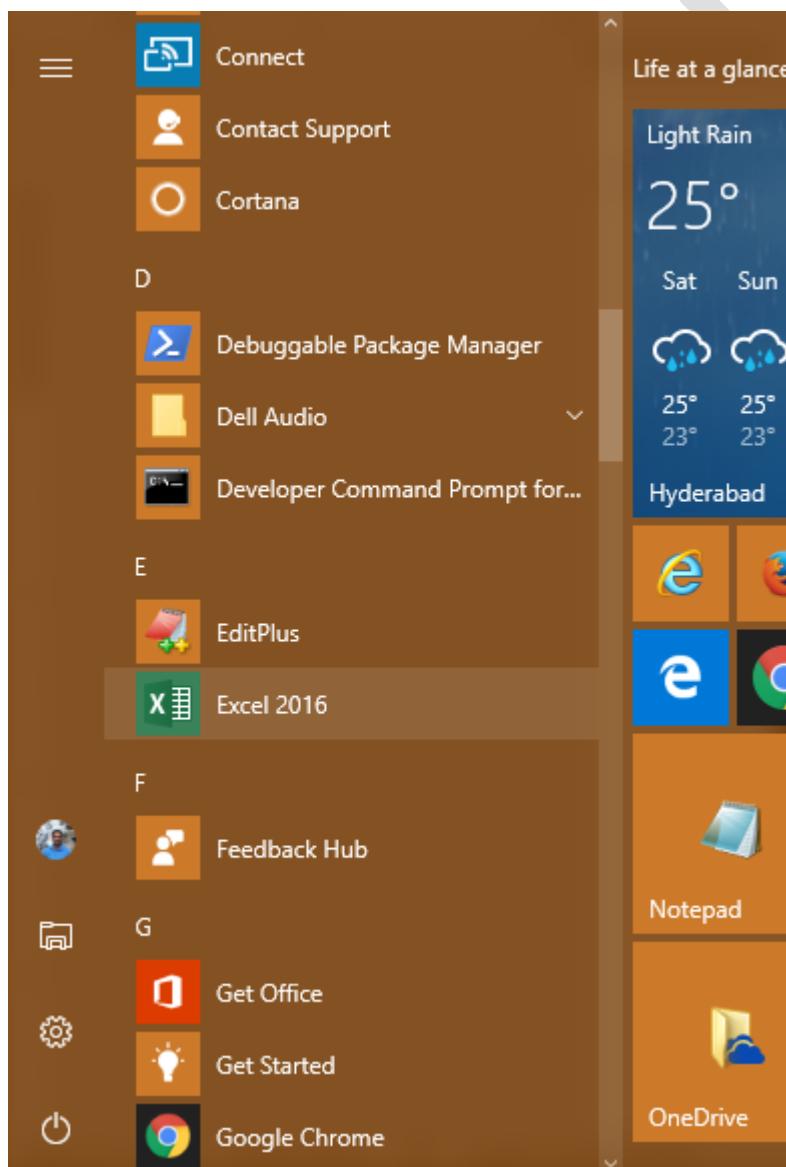
Installing Access Database Engine

- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at:
<https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>

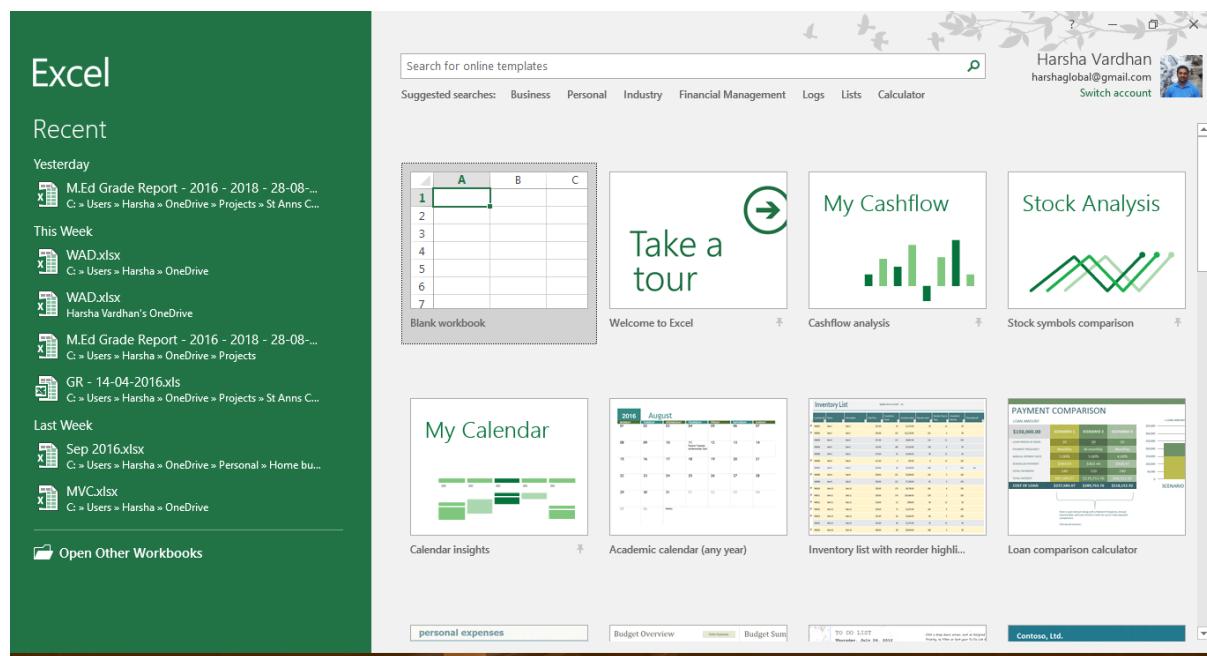
- Additionally, we have to install “Access Database Engine”, based on the steps explained in the previous examples.
- Note: Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>

Creating work book in MS Excel

- Go to “Start” – “Excel 2016”.



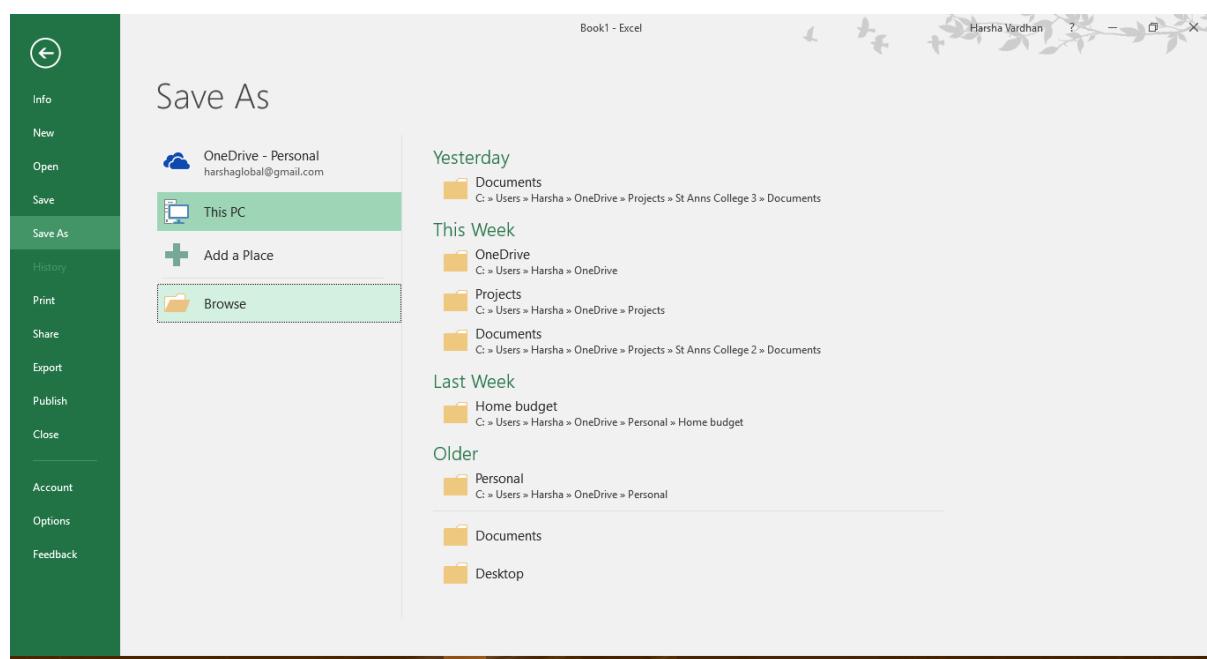
- Click on “Blank Workbook”.



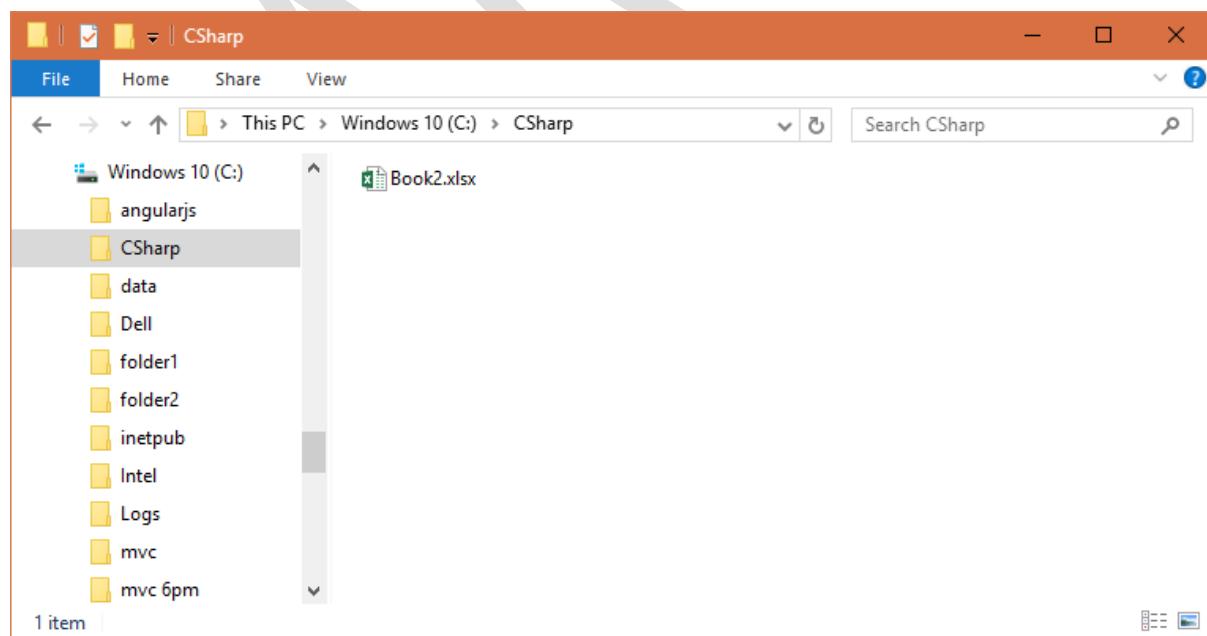
- Type the data as follows:

The screenshot shows a new Excel spreadsheet titled 'Book2.xlsx'. The first sheet is named 'Sheet1'. It has three columns: 'EmpID', 'EmpName', and 'Salary'. The 'EmpID' column is currently selected, indicated by a green border around its cells. The rest of the cells in the row are empty. The ribbon is visible at the top with the 'Home' tab selected. The 'Font' group shows 'Calibri' and '11'. The 'Cells' group shows 'AutoSum', 'Sort & Find & Select', and 'Clear'. The 'Styles' group shows 'Conditional Formatting', 'Table', 'Styles', and 'Format'. The 'Editing' group shows 'Insert', 'Delete', and 'Format'.

- Go to "File" – "Save" – "This PC" – "Browse".



- Select the folder as “C:\CSharp”.
- Type the filename as “Book2.xlsx”.
- Click on “Save”.
- Close “Microsoft Excel 2016”.
- Make sure “Book2.xlsx” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SqlServerToExcelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerToExcelExample”.
- Click on OK.

Program.cs

```
//Copy data from "SqlServer" to "Excel"  
using System;  
using System.Data.SqlClient;  
using System.Data.OleDb;  
using System.Data;  
  
namespace SqlServerToExcelExample  
{  
    class Program  
    {  
        static void Main()  
        {  
            /* create reference variables */  
            SqlConnection cn;  
            SqlCommand cmd;  
            SqlDataAdapter adp;  
            DataSet ds;  
            DataTable dt;  
            DataRow drow;
```

```

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);

    InsertIntoExcel(eid, ename, sal);
}

Console.WriteLine(dt.Rows.Count + " records copied");

```

```
Console.ReadKey();
}

private static void InsertIntoExcel(int eid, string ename, decimal sal)
{
    //create reference variables
    OleDbConnection cn;
    OleDbCommand cmd;

    //create objects
    cn = new OleDbConnection();
    cmd = new OleDbCommand();

    //calling properties
    cn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\CSharp\Book2.xlsx; Extended Properties='Excel
12.0;HDR=Yes;IMEX=3'";
    cmd.CommandText = string.Format("insert into [Sheet1$] values('{0}',
'{1}', '{2}')", eid, ename, sal);
    cmd.Connection = cn;

    //calling methods
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
}
}
```

Running the Project

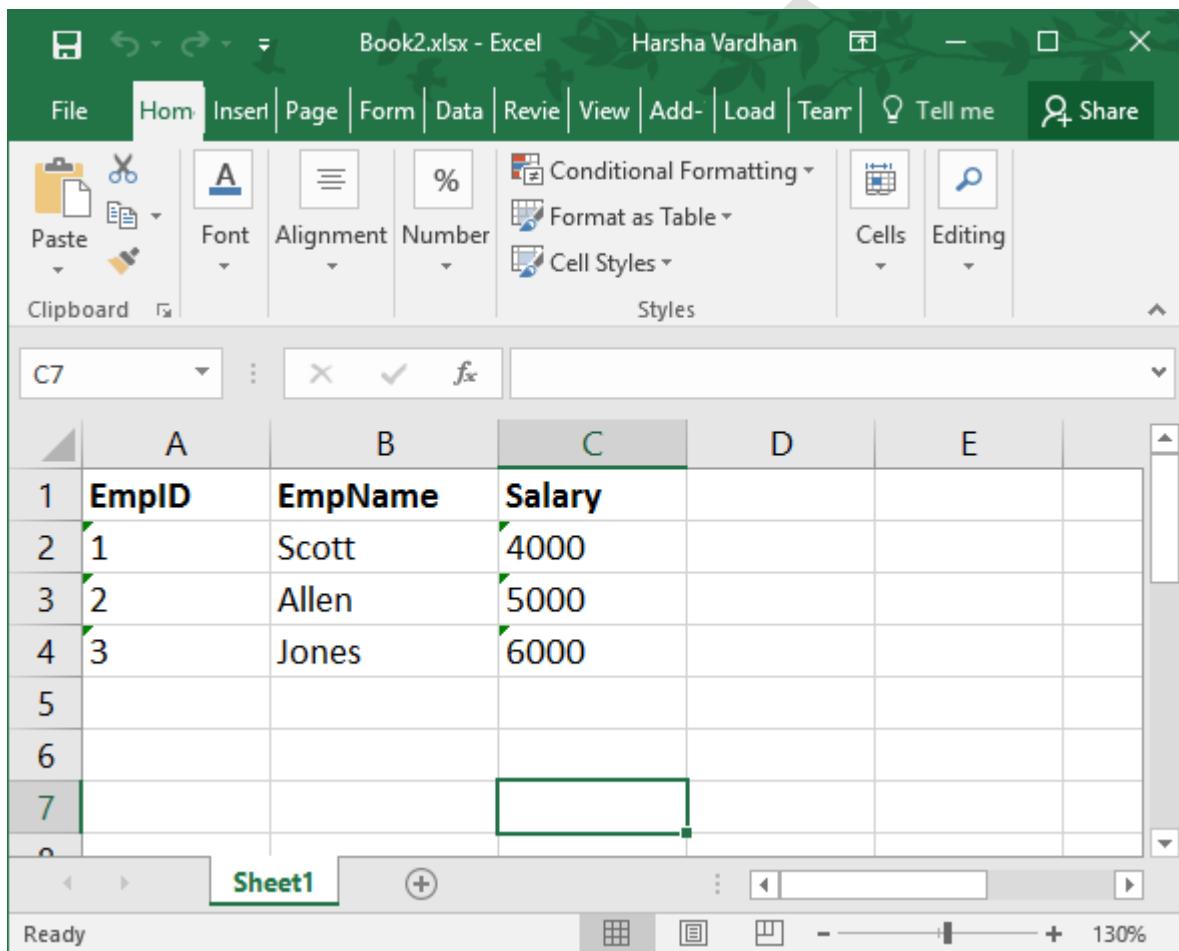
- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/304. Console - ADO.NET - SqlServer to Excel/Sq... - □ X
3 records copied
```

Note: If any database connection problem, it shows exception (run time error).

- Check the data in “C:\CSharp\Book2.xlsx” file.



The screenshot shows a Microsoft Excel window titled "Book2.xlsx - Excel". The ribbon menu is visible with tabs like File, Home, Insert, Page, Form, Data, Review, View, Add, Load, Team, Tell me, and Share. The "Home" tab is selected. The toolbar includes buttons for Paste, Font, Alignment, Number, Conditional Formatting, Format as Table, Cell Styles, Cells, and Editing. The formula bar shows "C7". The main area displays a table with four columns: EmpID, EmpName, and Salary. Row 1 contains the column headers. Rows 2, 3, and 4 contain data for employees with IDs 1, 2, and 3 respectively. Row 7 is empty. The table has a green border. The status bar at the bottom shows "Sheet1", zoom levels, and a 130% scale.

	A	B	C	D	E
1	EmpID	EmpName	Salary		
2	1	Scott	4000		
3	2	Allen	5000		
4	3	Jones	6000		
5					
6					
7					

ADO.NET – Oracle to SQL Server - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company  
go
```

```
use company  
go
```

```
create table Employees2(  
EmpID int,  
EmpName nvarchar(max),  
Salary decimal)  
go
```

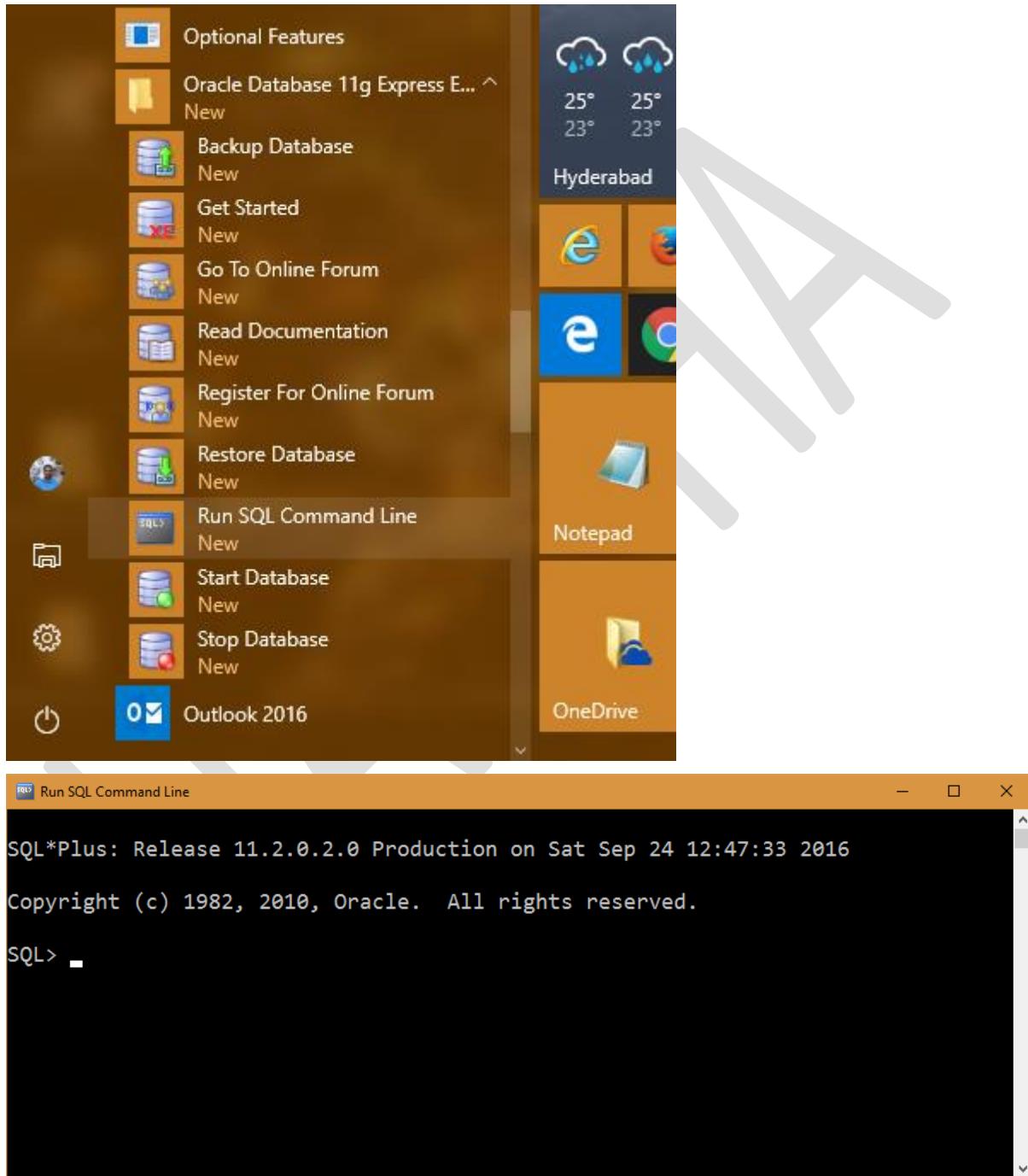
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Installing Oracle Database 11g Expression Edition

- Install “Oracle 11g Express” as shown in the previous example.
- You can download Oracle 11g Express Edition at:
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

Creating table in Oracle

- Note: Ignore this step, if you have created “employees2” table in Oracle already.
- Go to “Start” – “Oracle 11g Express Edition” – “Run SQL Command Line”.



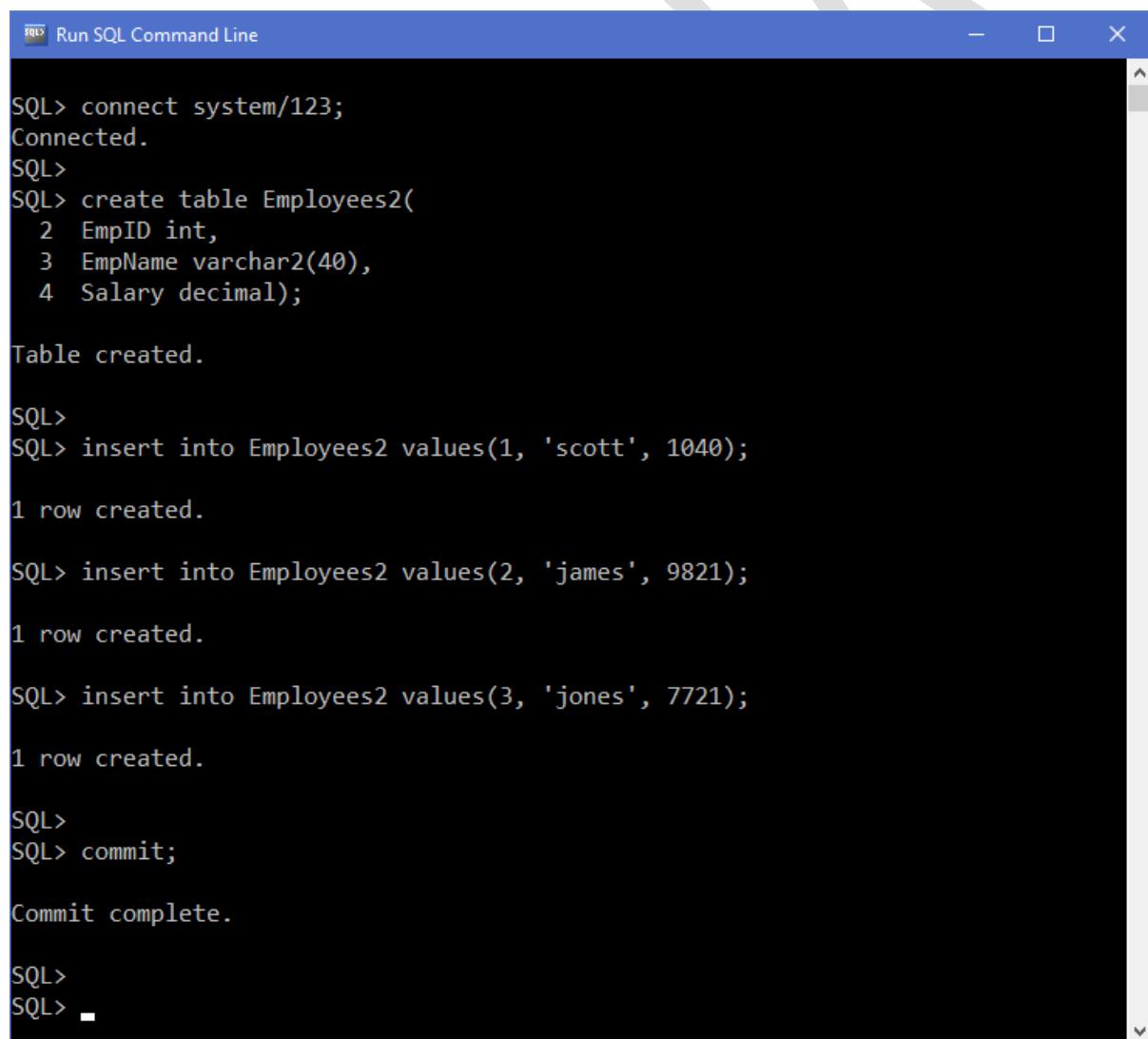
- Type the following script and press Enter.

connect system/123;

```
create table Employees2(
    EmpID int,
    EmpName varchar2(40),
    Salary decimal);

insert into Employees2 values(1, 'scott', 1040);
insert into Employees2 values(2, 'james', 9821);
insert into Employees2 values(3, 'jones', 7721);

commit;
```



The screenshot shows a Windows application window titled "Run SQL Command Line". The window contains a black text area where SQL commands are being run. The commands are identical to those shown in the text block above, resulting in the creation of a table named "Employees2" and the insertion of three rows of data. The output shows the table was created, three rows were inserted, and the commit was successful.

```
SQL> connect system/123;
Connected.
SQL>
SQL> create table Employees2(
2   EmpID int,
3   EmpName varchar2(40),
4   Salary decimal);

Table created.

SQL>
SQL> insert into Employees2 values(1, 'scott', 1040);

1 row created.

SQL> insert into Employees2 values(2, 'james', 9821);

1 row created.

SQL> insert into Employees2 values(3, 'jones', 7721);

1 row created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> ■
```

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “OracleToSqlServerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OracleToSqlServerExample”.
- Click on OK.

Program.cs

```
//Copy data from "Oracle" to "SqlServer"
using System;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;

namespace OracleToSqlServerExample
{
    class Program
    {
        static void Main()
        {
            OleDbConnection cn;
            OleDbCommand cmd;
            OleDbDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            cn = new OleDbConnection();
            cmd = new OleDbCommand();
            adp = new OleDbDataAdapter();
            ds = new DataSet();
        }
    }
}
```

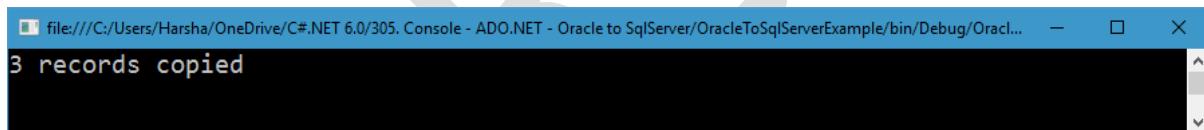
```
cn.ConnectionString = "user id=system; password=123;  
provider=msdaora.1";  
cmd.CommandText = "select * from Employees2";  
cmd.Connection = cn;  
adp.SelectCommand = cmd;  
  
adp.Fill(ds);  
dt = ds.Tables[0];  
for (int i = 0; i < dt.Rows.Count; i++)  
{  
    drow = dt.Rows[i];  
    object obj1, obj2, obj3;  
    obj1 = drow["EmpID"];  
    obj2 = drow["EmpName"];  
    obj3 = drow["Salary"];  
    int eid;  
    string ename;  
    decimal sal;  
    eid = Convert.ToInt32(obj1);  
    ename = Convert.ToString(obj2);  
    sal = Convert.ToDecimal(obj3);  
    InsertIntoSqlServer(eid, ename, sal);  
}  
Console.WriteLine(dt.Rows.Count + " records copied");  
Console.ReadKey();  
}  
  
private static void InsertIntoSqlServer(int eid, string ename, decimal  
sal)  
{  
    SqlConnection cn;  
    SqlCommand cmd;  
  
    cn = new SqlConnection();  
    cmd = new SqlCommand();
```

```
cn.ConnectionString = "data source=localhost; integrated  
security=yes; initial catalog=company";  
cmd.CommandText = string.Format("insert into Employees2  
values({0}, '{1}', {2})", eid, ename, sal);  
cmd.Connection = cn;  
cn.Open();  
cmd.ExecuteNonQuery();  
cn.Close();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



A screenshot of a Windows Command Prompt window. The title bar reads "file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/305. Console - ADO.NET - Oracle to SqlServer/OracleToSqlServerExample/bin/Debug/Orac...". The main window contains the text "3 records copied".

Note: If any database connection problem, it shows exception (run time error).

Check the data in “SQL Server”:

C#.NET 8.0

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'company' database is selected, showing its tables, including 'Employees' and 'Employees2'. A query window titled 'SQLQuery1.sql - HARSHA-PC\Harsha (51)*' displays the following SQL query:

```
select * from Employees2
```

The results pane shows the following data:

	EmpID	EmpName	Salary
1	1	scott	1040
2	2	james	9821
3	3	jones	7721

A status bar at the bottom indicates 'Query executed successfully.' and provides session details: HARSHA-PC (13.0 RC2) | HARSHA-PC\Harsha (51) | company | 00:00:00 | 3 rows.

ADO.NET – Excel to SQL Server - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
```

```
use company
go
```

```
create table Employees2(
EmpID int,
EmpName nvarchar(max),
Salary decimal)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

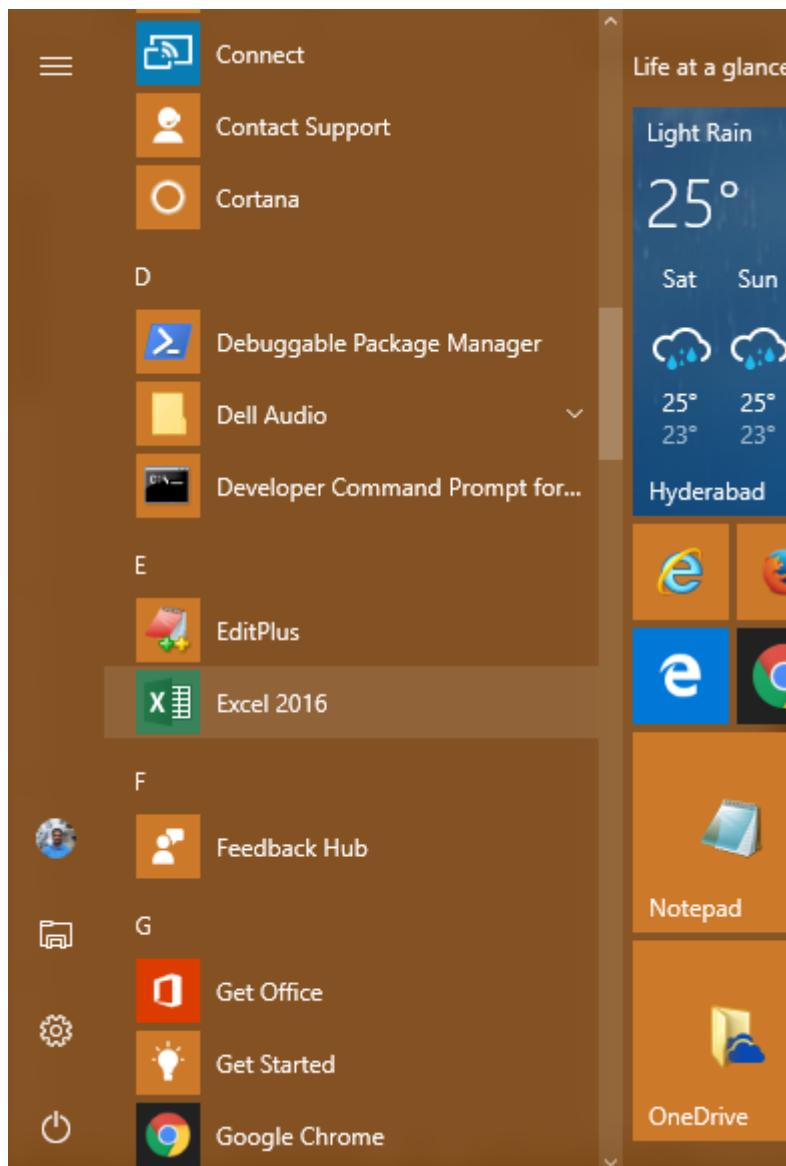
Installing Access Database Engine

- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at:
<https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”, based on the steps explained in the previous examples.
- Note: Ignore this step, if you have installed “Access Database Engine” already.

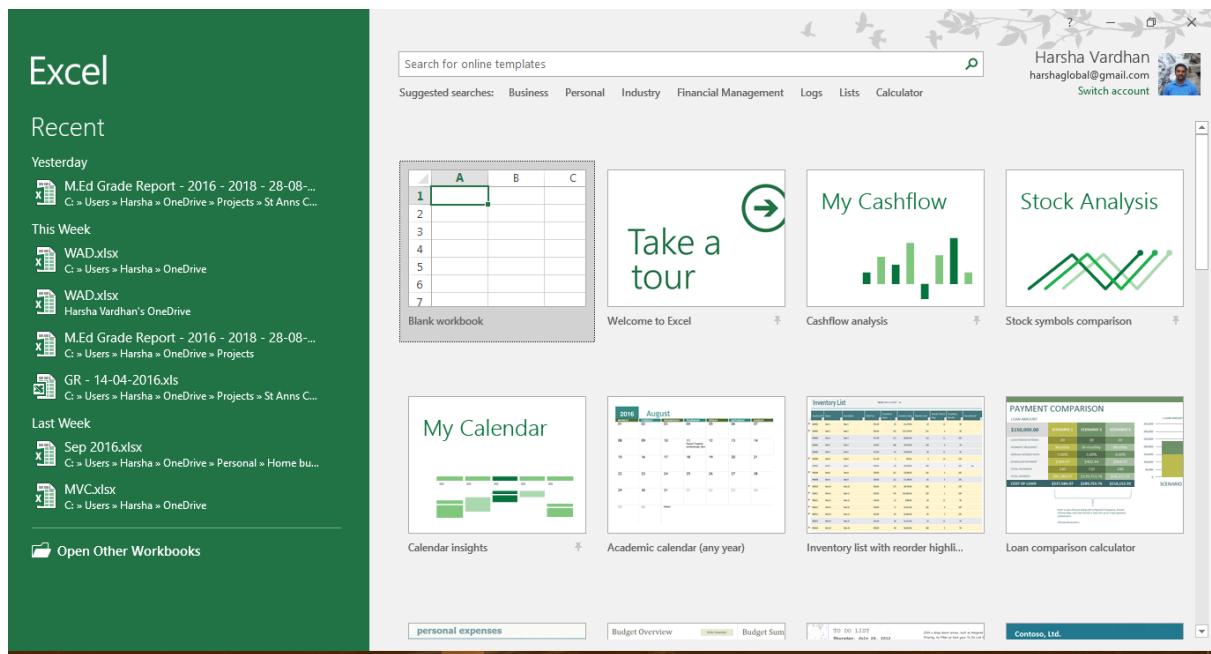
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>

Creating Workbook in “Excel”:

- Go to “Start” – “Excel 2016”.



- Click on “Blank Workbook”.

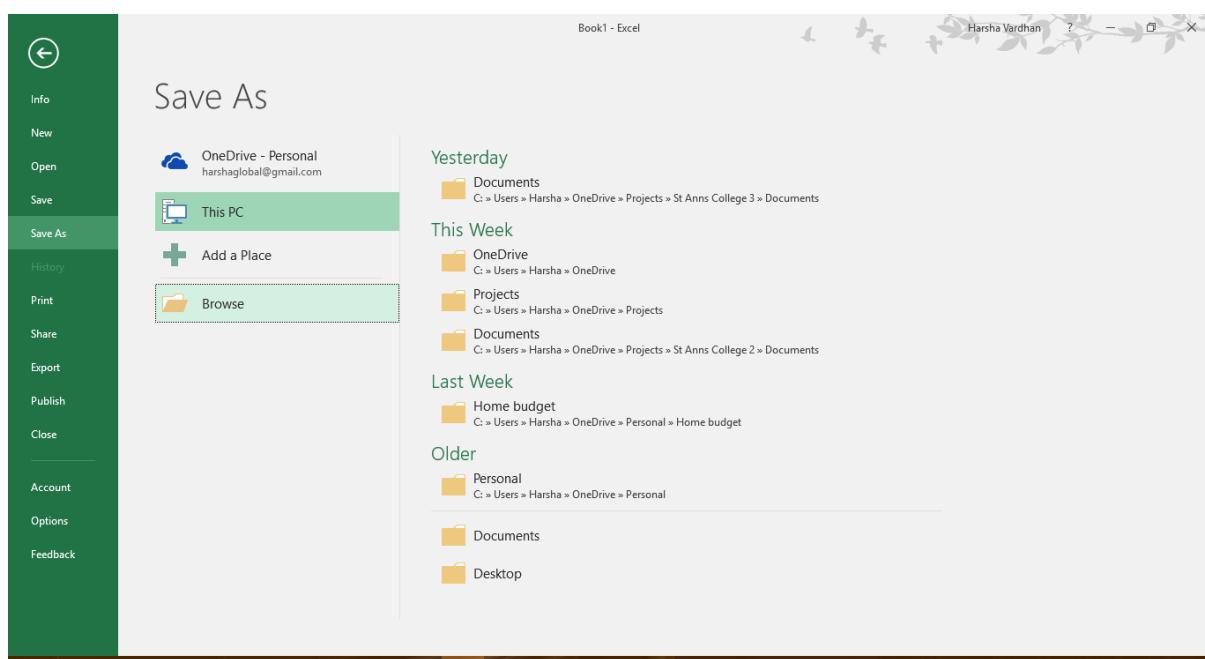


- Type the data as follows:

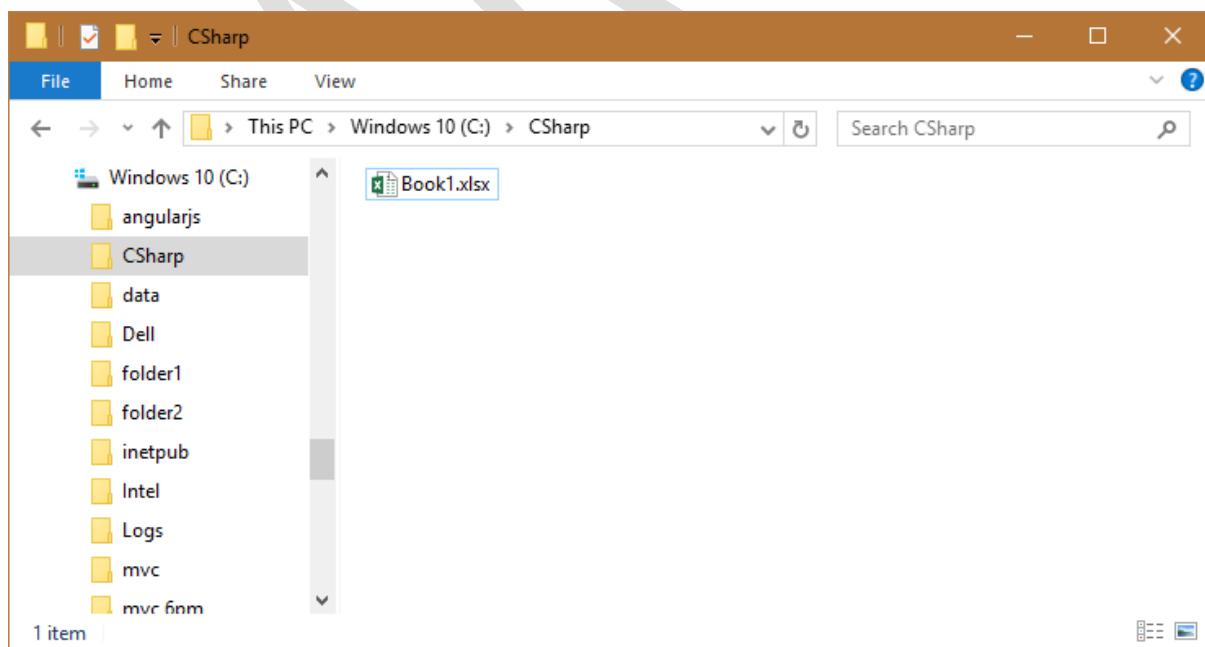
The screenshot shows a Microsoft Excel 2016 spreadsheet titled 'Book1 - Excel'. The table has columns labeled 'EmplID', 'EmpName', and 'Salary'. The data is as follows:

	EmplID	EmpName	Salary
1	1	abc	1000
2	2	def	2000
3	3	ghi	3000
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

- Go to "File" – "Save" – "This PC" – "Browse".



- Select the folder as “C:\CSharp”.
- Type the filename as “Book1.xlsx”.
- Click on “Save”.
- Close “Microsoft Excel 2016”.
- Make sure “Book1.xlsx” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “ExcelToSqlServerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExcelToSqlServerExample”.
- Click on OK.

Program.cs

```
//Copy data from "Excel" to "SqlServer"  
using System;  
using System.Data.SqlClient;  
using System.Data.OleDb;  
using System.Data;  
  
namespace ExcelToSqlServerExample  
{  
    class Program  
    {  
        static void Main()  
        {  
            /* create reference variables */  
            OleDbConnection cn;  
            OleDbCommand cmd;  
            OleDbDataAdapter adp;  
            DataSet ds;  
            DataTable dt;  
            DataRow drow;
```

```

/* create objects */
cn = new OleDbConnection();
cmd = new OleDbCommand();
adp = new OleDbDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\CSharp\Book1.xlsx; Extended Properties='Excel
12.0;HDR=Yes;IMEX=1'";
cmd.CommandText = "select * from [Sheet1$]";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);

    InsertIntoSqlServer(eid, ename, sal);
}

```

```
Console.WriteLine(dt.Rows.Count + " records copied");
Console.ReadKey();
}

private static void InsertIntoSqlServer(int eid, string ename, decimal
sal)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = string.Format("insert into Employees2
values({0}, '{1}', {2})", eid, ename, sal);
    cmd.Connection = cn;

    //calling methods
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/306. Console - ADO.NET - Excel to SqlServer/ExcelToSqlServerExample/bin/Debug/ExcelToS...
3 records copied
```

Note: If any database connection problem, it shows exception (run time error).

Check the data in “SQL Server”:

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays a tree view of databases, including 'company' which contains tables like 'Employees' and 'Employees2'. In the center, a query window titled 'SQLQuery1.sql - HARSHA-PC\Harsha (51)*' contains the SQL command: 'select * from Employees2'. To the right, the 'Results' tab shows the output of the query in a grid format:

EmplID	EmpName	Salary
1	scott	4466
2	allen	7562
3	jones	8833

Below the grid, a status bar indicates: 'Query executed successfully.' and 'HARSHA-PC (13.0 RC2) | HARSHA-PC\Harsha (51) | company | 00:00:00 | 3 rows'.

ADO.NET – SQL Server to File - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees2(
```

```
EmpID int,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “SqlServerToFileExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerToFileExample”. Click on OK.

Program.cs

```
//Copy data from "SqlServer" to "File"
using System;
using System.Data.SqlClient;
using System.Data;
using System.Diagnostics;
using System.IO;
namespace SqlServerToFileExample
{
    class Program
    {
        static void Main()
        {
            //create directory if not exists
            if (Directory.Exists(@"C:\CSharp") == false)
            {
                Directory.CreateDirectory(@"C:\CSharp");
            }

            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;
            FileInfo finfo;
            FileStream fs;
            StreamWriter sw;

            //create objects
            cn = new SqlConnection();
```

```

cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();
finfo = new FileInfo(@"C:\CSharp\Employees.txt");

//delete the file if already exists
if (finfo.Exists == true)
    finfo.Delete();

//create objects
fs = new FileStream(@"C:\CSharp\Employees.txt", FileMode.Create,
FileAccess.Write);
sw = new StreamWriter(fs);

//call properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees2";
cmd.Connection = cn;
adp.SelectCommand = cmd;

//call methods
adp.Fill(ds);

//read data
dt = ds.Tables[0];

//read data from database record-by-record
for (int i = 0; i < dt.Rows.Count; i++)
{
    //get values
    object obj1, obj2, obj3;
    drow = dt.Rows[i];
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    int eid = Convert.ToInt32(obj1);
}

```

```
        string ename = Convert.ToString(obj2);
        double sal = Convert.ToDouble(obj3);
        string msg = eid + "," + ename + "," + sal;
        //write data to the file
        sw.WriteLine(msg);
    }

    //close the file
    sw.Close();

    Console.WriteLine("Data has been written to the following file
successfully.\nC:\\CSharp\\Employees.txt");

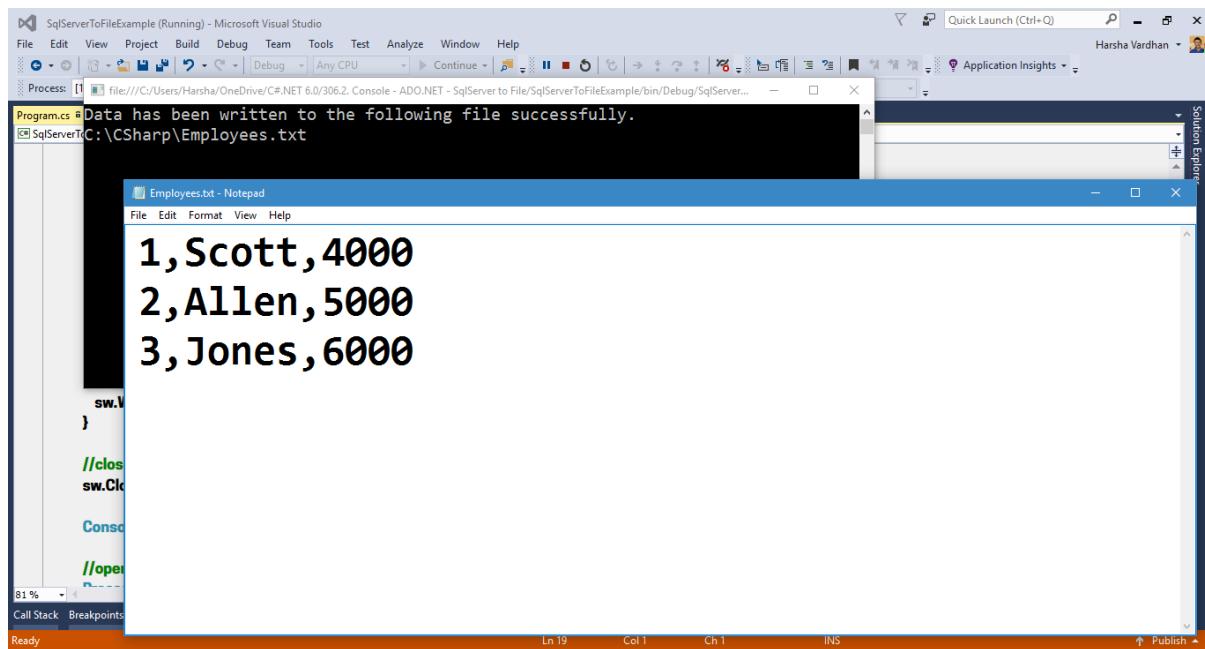
    //open the file in notepad
    Process.Start(@"C:\\Windows\\System32\\notepad.exe",
 @"C:\\CSharp\\Employees.txt");

    Console.ReadKey();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



SqlServerToFileExample (Running) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/3062_Console - ADO.NET - SqlServer to File/SqlServerToFileExample/bin/Debug/SqlServer...]

Program.cs Data has been written to the following file successfully.

SqlServerToFileExample C:\CSharp\Employees.txt

Employees.txt - Notepad

File Edit Format View Help

1, Scott, 4000
2, Allen, 5000
3, Jones, 6000

sw.WriteLine("1, Scott, 4000");
sw.WriteLine("2, Allen, 5000");
sw.WriteLine("3, Jones, 6000");
sw.Close();
Console.WriteLine("Data has been written to the following file successfully.");
Console.ReadLine();

Call Stack Breakpoints

Ready Ln 19 Col 1 Ch 1 INS ↑ Publish ↓

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – File to SQL Server - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees2(
EmpID int,
EmpName nvarchar(max),
Salary decimal)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.
- Type the project name as “FileToSqlServerExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “FileToSqlServerExample”.
- Click on OK.

Program.cs

```
//Copy data from "File" to "SqlServer"
using System;
using System.Data.SqlClient;
using System.IO;

namespace FileToSqlServerExample
{
    class Program
    {
        static void Main()
        {
            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            FileInfo finfo;
            FileStream fs;
            StreamReader sr;
            SqlParameter p1, p2, p3;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
            p2 = new SqlParameter();
            p3 = new SqlParameter();
            finfo = new FileInfo(@"C:\CSharp\Employees.txt");

            //check whether the file already exists or not
            if (finfo.Exists == false)
            {
                Console.WriteLine("File not exists.");
            }
        }
    }
}
```

```

        Console.ReadKey();
        return;
    }

    //create objects
    fs = new FileStream(@"C:\CSharp\Employees.txt", FileMode.Open,
FileAccess.Read);
    sr = new StreamReader(fs);

    //call properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "insert into Employees2 values(@empid,
@empname, @salary)";
    cmd.Connection = cn;

    //call methods
    cmd.Parameters.Add(p1);
    cmd.Parameters.Add(p2);
    cmd.Parameters.Add(p3);

    //read all lines from the file
    while (true)
    {
        string line = sr.ReadLine();
        string[] values;
        values = line.Split(',');

        int eid = Convert.ToInt32(values[0]);
        string ename = Convert.ToString(values[1]);
        double sal = Convert.ToDouble(values[2]);

        p1.ParameterName = "@empid";
        p1.Value = eid;
        p2.ParameterName = "@empname";
        p2.Value = ename;
        p3.ParameterName = "@salary";
    }
}

```

```

p3.Value = sal;

//insert into database
cn.Open();
cmd.ExecuteNonQuery();
cn.Close();

if (sr.EndOfStream == true)
    break;
}

//close the file
sr.Close();

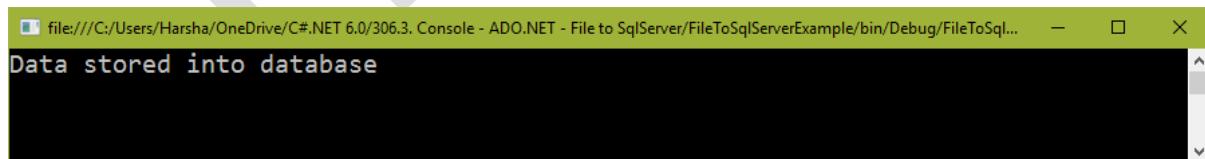
Console.WriteLine("Data stored into database");
Console.ReadKey();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Note: If any database connection problem, it shows exception (run time error).

Check the data in “SQL Server”:

C#.NET 8.0

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'company' database is selected, showing its structure with tables like 'Employees' and 'Employees2'. A query window titled 'SQLQuery1.sql - HARSHA-PC\Harsha (52)*' displays the results of the query 'select * from Employees2'. The results grid shows three rows of data:

	EmpID	EmpName	Salary
1	1	Scott	4000
2	2	Allen	5000
3	3	Jones	6000

A status bar at the bottom indicates 'Query executed successfully.' and other session details.

The "SqlCommandBuilder" class in ADO.NET

System.Data.SqlClient.SqlCommandBuilder

- The “SqlCommandBuilder” is a pre-defined class in a namespace called “System.Data.SqlClient”.
- This class’s object generates a set of INSERT, UPDATE, DELETE sql statements automatically, based on the SELECT statement, based on which we can perform DML operations using DataSet.

Properties of “SqlCommandBuilder” class:

Sl. No	Property	Data Type	Description
1	DataAdapter	SqlDataAdapter	<p>Represents the reference of object of SqlDataAdapter class in which the INSERT, UPDATE, DELETE commands have to be generated..</p> <p><u>Syntax:</u> <i>referencevariable</i>.DataAdapter = <i>ReferenceVariableSqlDataAdapter</i>;</p>

Constructors of “SqlCommandBuilder” class:

Sl. No	Constructor	Description
1	SqlCommandBuilder()	<p>Initializes nothing.</p> <p><u>Syntax:</u> new SqlCommandBuilder();</p>
2	SqlCommandBuilder (SqlDataAdapter adapter)	<p>Initializes DataAdapter property.</p> <p><u>Syntax:</u> new SqlCommandBuilder(SqlDataAdapter adapter);</p>

ADO.NET – SqlCommandBuilder – DataSet – Insertion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “DataSetInsertionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetInsertionExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace DataSetInsertionExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());

            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            SqlCommandBuilder cmb;
            DataSet ds;
            DataTable dt;
            DataRow drow;
```

```
//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
cmb = new SqlCommandBuilder();
ds = new DataSet();

//call properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;
cmbDataAdapter = adp;

//call methods
adp.Fill(ds);
dt = ds.Tables[0];

//create a new data row
drow = dt.NewRow();
drow["empid"] = empId;
drow["empname"] = empName;
drow["salary"] = sal;

//add row to the table
dt.Rows.Add(drow);

//save changes back to the database
adp.Update(ds);

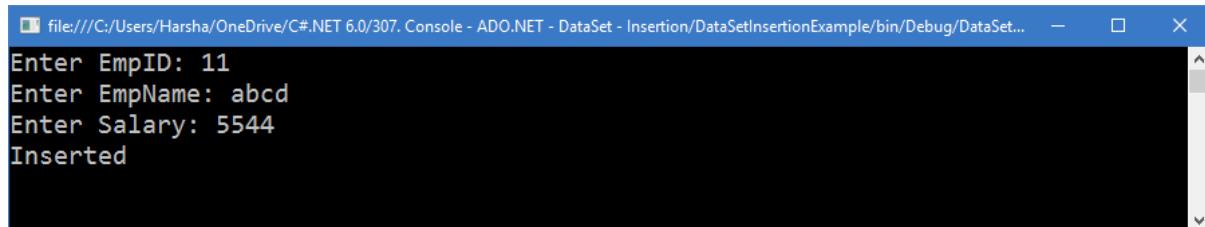
Console.WriteLine("Inserted");
Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/307. Console - ADO.NET - DataSet - Insertion/DataSetInsertionExample/bin/Debug/DataSet... - X
Enter EmpID: 11
Enter EmpName: abcd
Enter Salary: 5544
Inserted
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – SqlCommandBuilder – DataSet – Updation - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “DataSetUpdationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetUpdationExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace DataSetUpdationExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());

            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            SqlCommandBuilder cmb;
            DataSet ds;
            DataTable dt;
            DataRow drow;
```

```
//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
cmb = new SqlCommandBuilder();
ds = new DataSet();

//call properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;
cmbDataAdapter = adp;

//call methods
adp.Fill(ds);
dt = ds.Tables[0];

//get datarow based on empid
int n = Convert.ToInt32(empId);
string condition = "empid=" + n;
drow = dt.Select(condition)[0];

//update
drow["empname"] = empName;
drow["salary"] = sal;

//save changes back to the database
adp.Update(ds);

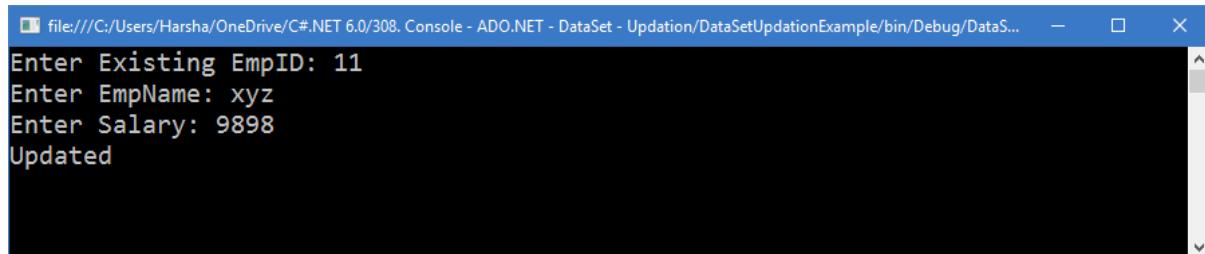
Console.WriteLine("Updated");
Console.ReadKey();
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/308. Console - ADO.NET - DataSet - Updation/DataSetUpdationExample/bin/Debug/DataS...
Enter Existing EmpID: 11
Enter EmpName: xyz
Enter Salary: 9898
Updated
```

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – SqlCommandBuilder – DataSet – Deletion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “DataSetDeletionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetDeletionExample”.
- Click on OK.

Program.cs

```
using System;
using System.Data.SqlClient;
using System.Data;

namespace DataSetDeletionExample
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());

            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            SqlCommandBuilder cmb;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();
            adp = new SqlDataAdapter();
```

```

cmb = new SqlCommandBuilder();
ds = new DataSet();

//call properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;
cmbDataAdapter = adp;

//call methods
adp.Fill(ds);
dt = ds.Tables[0];

//get datarow based on empid
int n = Convert.ToInt32(emplId);
string condition = "empid=" + n;
drow = dt.Select(condition)[0];

//delete the row
drow.Delete();

//save changes back to the database
adp.Update(ds);

Console.WriteLine("Deleted");
Console.ReadKey();
}

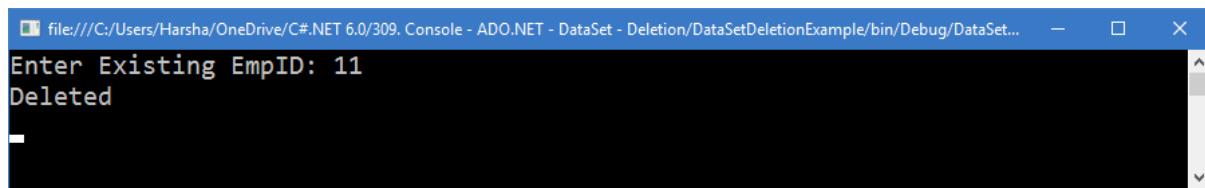
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/309. Console - ADO.NET - DataSet - Deletion/DataSetDeletionExample/bin/Debug/DataSet...
Enter Existing EmpID: 11
Deleted
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

N-Tier Architecture

N-Tier Architecture (or) 3-Tier Architecture

- The application's code is divided as 3 major parts for ease of maintenance and administration purpose:
 1. Data Access Layer: Contains code for connecting with database.
 2. Business Logic Layer: Contains code for validations and calculations.
 3. Presentation Layer: Contains code for displaying the UI and event handling.
- We have to create each "layer (part)" as a separate class library.

N-Tier Architecture - Example

Creating Database

- Note: Ignore this step, if you have created "company" database already.
- Open SQL Server Management Studio. Click on "Connect".
- Click on "New Query".
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
EmplID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Solution

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Other Project Types” – “Visual Studio Solutions”.
- Select “Blank Solution”.
- Type the name as “CompanySolution”.
- Type the location as “C:\CSharp”.
- Click on OK.

Creating Data Access Layer

- Open Solution Explorer.
- Right click on the solution (CompanySolution) and click on “Add” – “New Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Class Library”.
- Type the name as “CompanyDataAccess”.
- Click on OK.

Creating Business Logic Layer

- Open Solution Explorer.

- Right click on the solution (CompanySolution) and click on “Add” – “New Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Class Library”.
- Type the name as “CompanyBusinessLogic”.
- Click on OK.

Creating Presentation Logic

- Open Solution Explorer.
- Right click on the solution (CompanySolution) and click on “Add” – “New Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Console Application”.
- Type the name as “CompanyPresentationLogic”.
- Click on OK.

Adding references:

- Open Solution Explorer.
- Right click on “CompanyBusinessLogic” project and click on “Add” – “Reference” – “Projects” – Check the checkbox “CompanyDataAccess” – OK.
- Right click on “CompanyPresentationLogic” project and click on “Add” – “Reference” – “Projects” – Check the checkbox “CompanyBusinessLogic” – OK.

Setting the Startup Project

- Open Solution Explorer.
- Right click on “CompanyPresentationLogic” project and click on “Set as Startup Project”.

CompanyDataAccess\Class1.cs

```
//Data Access Logic
using System;
using System.Data.SqlClient;

namespace CompanyDataAccess
{
    public class EmployeesDataAccess
    {
        public int InsertEmployee(int EmpID, string EmpName, decimal Salary)
        {
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1, p2, p3;

            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
            p2 = new SqlParameter();
            p3 = new SqlParameter();

            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
            cmd.CommandText = "insert into Employees values(@a, @b, @c)";
            cmd.Connection = cn;
            p1.ParameterName = "@a";
            p2.ParameterName = "@b";
            p3.ParameterName = "@c";
            p1.Value = EmpID;
            p2.Value = EmpName;
            p3.Value = Salary;
            cmd.Parameters.Add(p1);
            cmd.Parameters.Add(p2);
            cmd.Parameters.Add(p3);
            cn.Open();
            int n = cmd.ExecuteNonQuery();
            cn.Close();
        }
    }
}
```

```
    return n;
}
}
}
```

CompanyBusinessLogic\Class1.cs

```
//Business Logic
using System;
using CompanyDataAccess;

namespace CompanyBusinessLogic
{
    public class EmployeesBusinessLogic
    {
        private int _empid;
        private string _empname;
        private decimal _salary;

        public int EmpID
        {
            set
            {
                if (value >= 0)
                {
                    _empid = value;
                }
            }
            get
            {
                return _empid;
            }
        }

        public string EmpName
        {
```

```
set
{
    if (value.Length <= 40)
    {
        _empname = value;
    }
}
get
{
    return _empname;
}

public decimal Salary
{
    set
    {
        if (value >= 1000 && value <= 5000)
        {
            _salary = value;
        }
    }
    get
    {
        return _salary;
    }
}

public string Insert()
{
    EmployeesDataAccess da = new EmployeesDataAccess();
    int result = da.InsertEmployee(this.EmpID, this.EmpName,
this.Salary);
    if (result == 1)
    {
        return "Inserted";
    }
}
```

```
        else
    {
        return "Error in insertion";
    }
}
}
```

CompanyPresentationLogic\Program.cs

```
//Presentation Logic
using System;
using CompanyBusinessLogic;

namespace CompanyPresentationLogic
{
    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());

            EmployeesBusinessLogic b = new EmployeesBusinessLogic();
            b.EmpID = empId;
            b.EmpName = empName;
            b.Salary = sal;
            string msg = b.Insert();
            Console.WriteLine(msg);
            Console.ReadKey();
        }
    }
}
```

}

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/316. Console - 3-Tier Architecture/CompanySolution/CompanySolution/bin/Debug/Com...
Enter EmpID: 21
Enter EmpName: sample
Enter Salary: 7000
Inserted
```

Note: If any database connection problem, it shows exception (run time error).

LINQ to SQL

- “LINQ to SQL” is a framework, which is used to connect and interact with SQL Server database.
- “LINQ to SQL” was introduced in .NET 3.5.
- “LINQ to SQL” internally uses “ADO.NET” and “SQL”. That means “LINQ to SQL queries” will be internally converted into “SQL queries” and executed based on ADO.NET classes.
- “LINQ to SQL” works based on mapping classes. A mapping class represents the structure of the table. A table become as “mapping class” and all the columns become as “Properties” in the “mapping class”.
- The “DataContext” class that inherits from “System.Data.Linq.DataContext” class, represents a group of collections of mapping classes.

Syntax of “LINQ to SQL”:

(from *rangevariable* in *collectionname* where *condition* select *rangevariable*).ToList()

-- or --

Collectionname.Where(*variable* => *condition*).ToList()

The ToList() method opens the connection, convert the LINQ query into SQL query, execute the query, convert the resultset into collection and close the database connection.

Advantages of LINQ to SQL

- Easier than “ADO.NET”.
- Converts data into collection; so we can write the loops easily.

LINQ to SQL - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

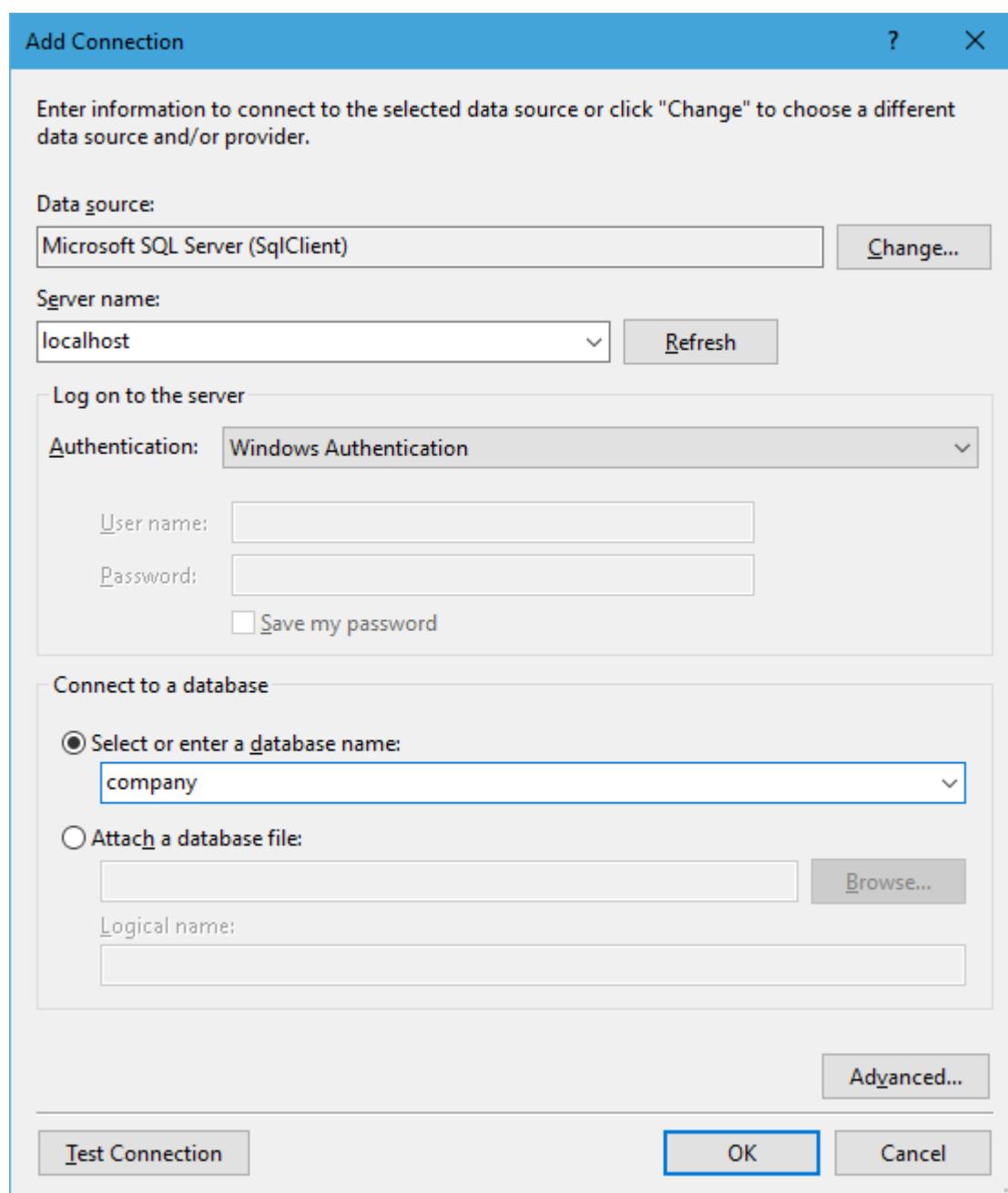
Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

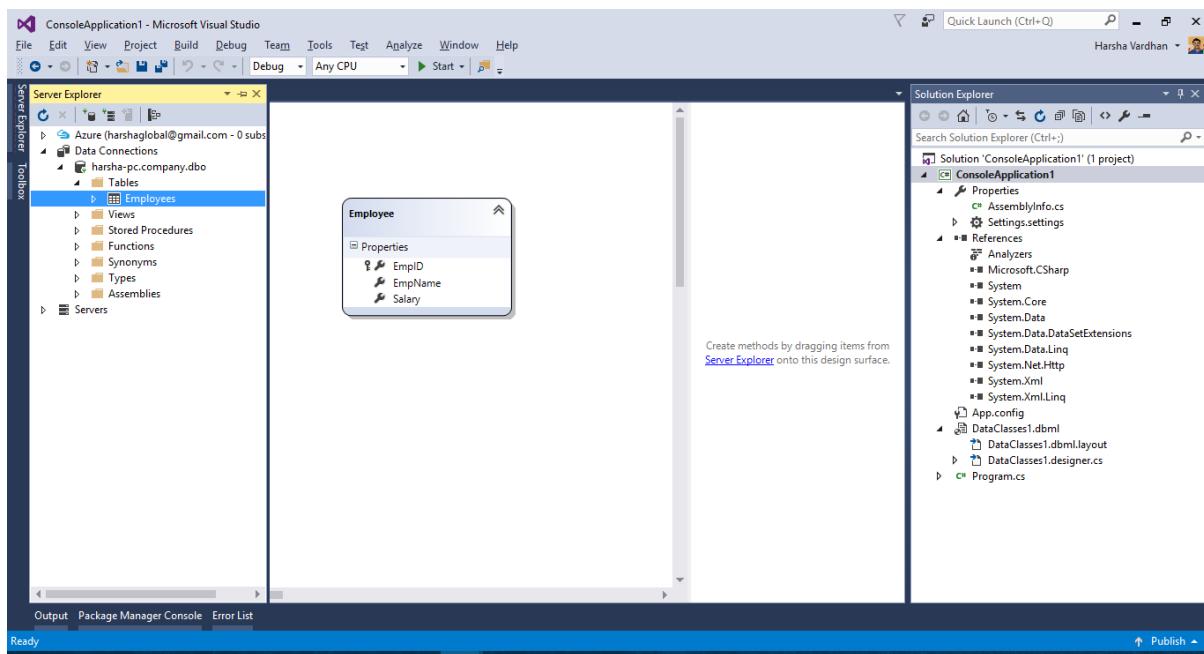
- Select “Console Application”.
- Type the project name as “LinqToSqlExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LinqToSqlExample”.
- Click on OK.

Creating “DBML” file:

- Open Solution Explorer.
- Right click on the project (LinqToSqlExample) and click on “Add” – “New Item” – “Data” – “LINQ to SQL Classes”.
- Type the name as “DataClasses1.dbml” (Database Markup Language).
- Click on “Add”.
- Go to “View” menu – “Server Explorer”.
- Right click on “Data Connections” and click on “Add Connection”.
- Type the server name as “localhost”.
- Select the authentication as “Windows Authentication”.
- Select the database name as “company”.
- Click on OK.



- Expand “company.dbo” – “Tables” – “Employees”.
- Drag and drop “Employees” table from Server Explorer into “Object Relational Designer” of “DataClasses1.dbml” file.
- It generates two classes called “DataClasses1DataContext” and “Employee”.



- Go to “Build” menu – “Build Solution”.

Program.cs

```

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

namespace LinqToSqlExample
{
    class Program
    {
        static void Main()
        {
            //create reference variables
            DataClasses1DataContext db;

            //create objects
            db = new DataClasses1DataContext();

            //LINQ query
            List<Employee> emps = db.Employees.ToList();
        }
    }
}

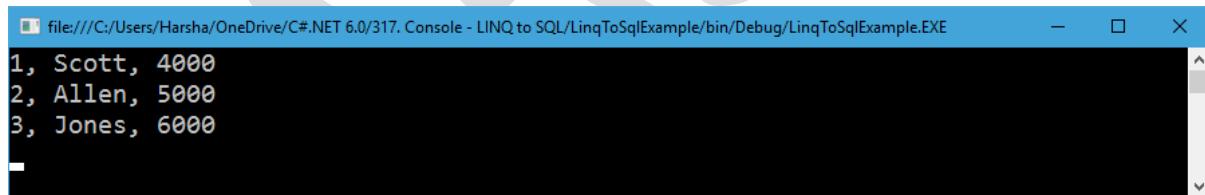
```

```
//read data row-by-row
for (int i = 0; i < emps.Count; i++)
{
    Employee emp = emps[i];
    Console.WriteLine(emp.EmpID + "," + emp.EmpName + "," +
emp.Salary);
}
Console.ReadKey();
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
1, Scott, 4000
2, Allen, 5000
3, Jones, 6000
```

Note: If any database connection problem, it shows exception (run time error).

C#.NET – Console – Entity Framework

Intro to ADO.NET Entity Framework

- “ADO.NET Entity Framework (EF)” is a framework, which is a part of “.NET”, which is advanced than “ADO.NET”. It was introduced in .NET 4.0.
- “ADO.NET EF” works based on “Ado.NET”.
- “ADO.NET EF” is a database technology, which is used to connect and interact with databases.

Advantages of Entity Framework

- “Entity Framework” is easier than “ADO.NET”.
- “Entity Framework” automatically converts data into collection; so that it will be easy to read and manipulate the data.
- “Entity Framework” is better compatible with other frameworks such as “WPF” and “ASP.NET MVC” etc.

Features of Entity Framework

1. Tables become “model classes”; columns become “properties”.

“Employees” table → “Employee” class

- “EmpID” column → “EmpID” property
- “EmpName” column → “EmpName” property
- “Salary” column → “Salary” property.

2. Every record (row) become “object”.

- EmplD: 1, EmpName: Scott, Salary: 4000 → Object
3. Every “group of objects” is a collection:
- A group of objects of model class, is treated as a “collection”.
4. We can write presentation logic / business logic based on the collections:
- We will read data from collection; not from the dataset / data reader.
 - ADO.NET EF internally and automatically copy the data from “resultset” to “collection”.

Steps for development of Entity Framework

1. Install “EntityFramework” package:
 - We have to install “EntityFramework” NuGet package in order to use Entity Framework.
 - A NuGet package is a collection of one or more assemblies. When we install a NuGet package, all of its assemblies will be added as reference to the project.
 - The “EntityFramework” NuGet package contains an assembly called “EntityFramework.dll”, which contains all the pre-defined classes of Entity Framework.
 - To install the “EntityFramework” package, use the following command in “Package Manager Console”.

Install-package EntityFramework

2. Create a model class:

```
class modelclassname
{
    public datatype property1 { get; set; }
    public datatype property2 { get; set; }
    ...
}
```

3. Create a child class for “DbContext” class:

DbContext = Group of DbSet's
DbSet = a collection of one table

```
class childclassname : System.Data.Entity.DbContext
{
    public DbSet<modelclass> collectionname { get; set; }
    public DbSet<modelclass> collectionname { get; set; }
    ...
}
```

4. Create an object for the child class:

```
Childclassname referencevariable ;
referencevariable = new Childclassname();
```

5. Access the collection:

Referencevariable.collectionname

Entity Framework - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

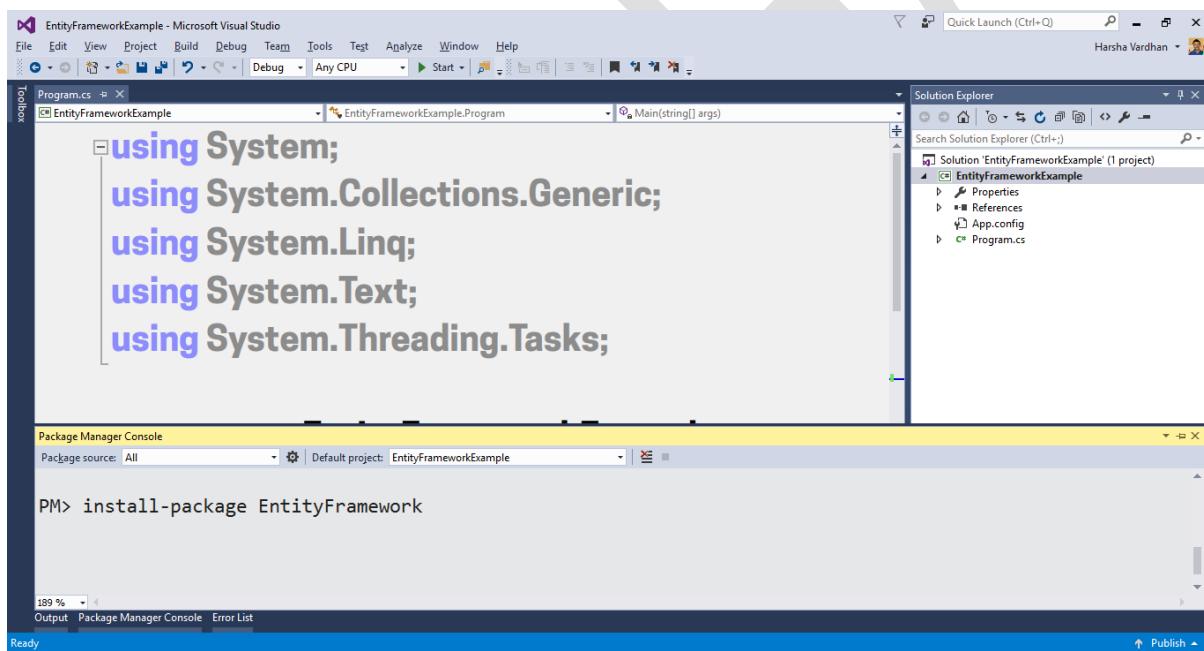
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “EntityFrameworkExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EntityFrameworkExample”.
- Click on OK.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

Install-package EntityFramework



Adding “ConnectionString” in “App.Config”:

- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

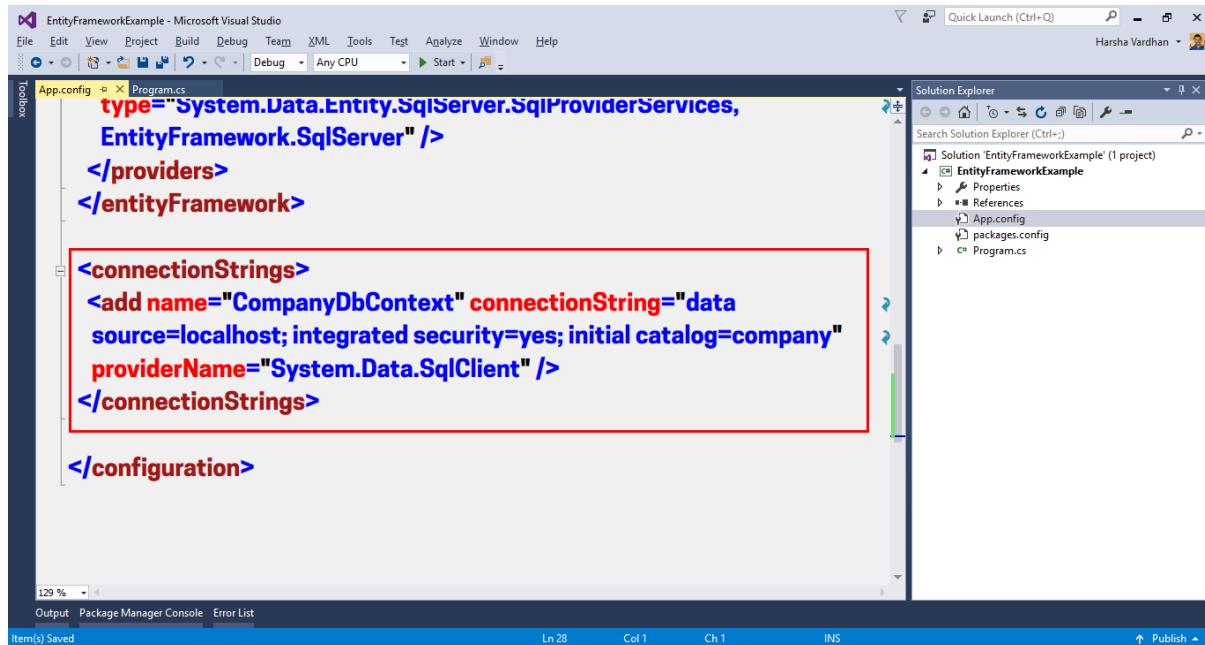
```

<connectionStrings>
  <add name="CompanyDbContext"
    connectionString="data source=localhost; integrated
  
```

```

    security=yes; initial catalog=company"
providerName="System.Data.SqlClient" />
</connectionStrings>

```



Program.cs

```

using System;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
using System.Collections.Generic;
using System.Linq;

namespace EntityFrameworkExample
{
    public class Employee
    {
        [Key]
        public int EmpID { get; set; }
        public string EmpName { get; set; }
        public decimal Salary { get; set; }
    }
}

```

```

public class CompanyDbContext : DbContext
{
    public DbSet<Employee> Employees { get; set; }
}

class Program
{
    static void Main()
    {
        //create reference variable
        CompanyDbContext db;
        //create object
        db = new CompanyDbContext();

        //LINQ query
        //List<Employee> emps = db.Employees.ToList();
        //List<Employee> emps = db.Employees.Where(temp => temp.Salary
        >= 2000).ToList();
        List<Employee> emps = db.Employees.OrderBy(temp =>
        temp.Salary).ToList();

        //read data row-by-row
        for (int i = 0; i < emps.Count; i++)
        {
            Employee emp = emps[i];
            Console.WriteLine(emp.EmpID + "," + emp.EmpName + "," +
            emp.Salary);
        }
        Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/319. Console - EntityFramework - LINQ/1/EntityFrameworkExample/EntityFrameworkExam... — X
1, Scott, 4000
2, Allen, 5000
3, Jones, 6000
```

Note: If any database connection problem, it shows exception (run time error).

Entity Framework – FirstOrDefault - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

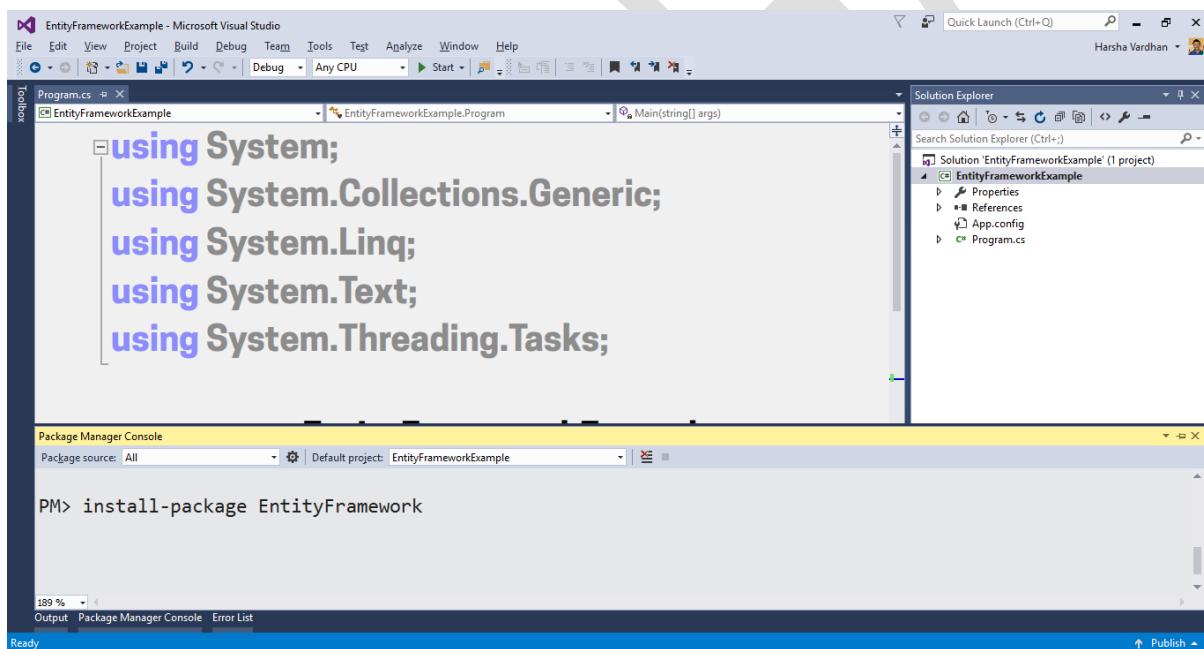
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Console Application”.

- Type the project name as “FirstOrDefaultExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FirstOrDefaultExample”.
- Click on OK.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

Install-package EntityFramework

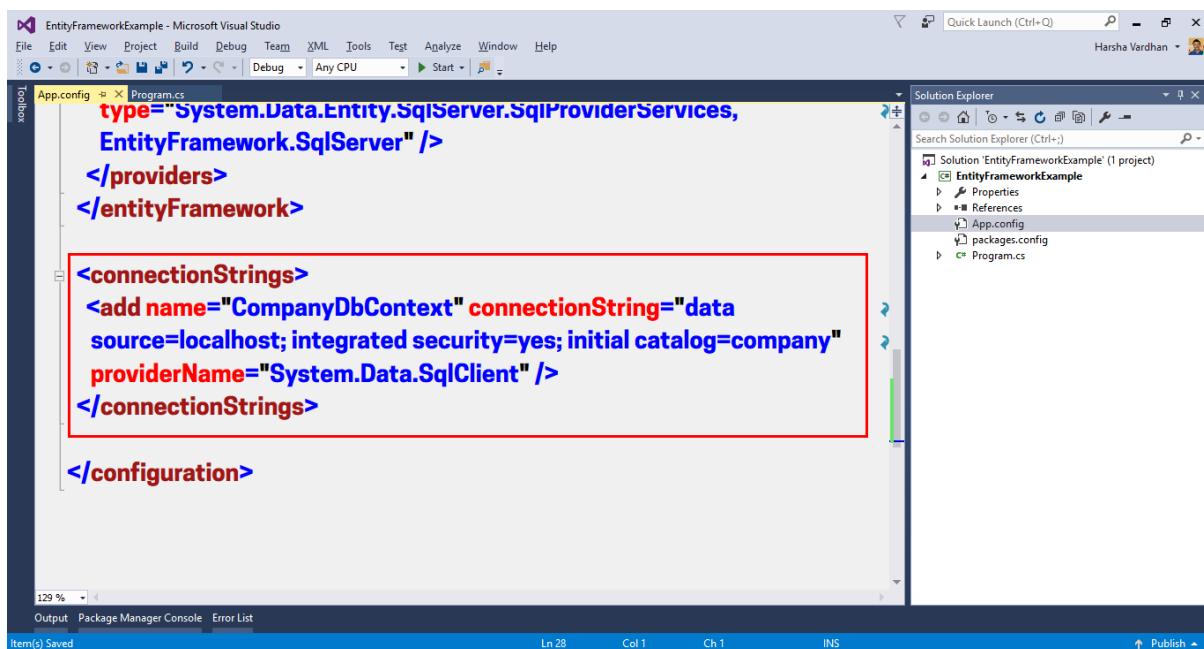


Adding “ConnectionString” in “App.Config”:

- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

```
<connectionStrings>
  <add name="CompanyDbContext" connectionString="data
  source=localhost; integrated security=yes; initial
  catalog=company" providerName="System.Data.SqlClient" />
```

</connectionStrings>



```

<configuration>
  <connectionStrings>
    <add name="CompanyDbContext" connectionString="data
      source=localhost; integrated security=yes; initial catalog=company"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>

```

The screenshot shows the Microsoft Visual Studio interface with the 'App.config' file open. The 'connectionStrings' section is highlighted with a red box. The configuration file contains the following XML:

Program.cs

```

using System;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
using System.Collections.Generic;
using System.Linq;

namespace FirstOrDefaultExample
{
  public class Employee
  {
    [Key]
    public int EmpID { get; set; }
    public string EmpName { get; set; }
    public decimal Salary { get; set; }
  }

  public class CompanyDbContext : DbContext

```

```
{  
    public DbSet<Employee> Employees { get; set; }  
}  
  
class Program  
{  
    static void Main()  
    {  
        //create reference variables  
        CompanyDbContext db;  
  
        //create objects  
        db = new CompanyDbContext();  
  
        //LINQ query: Get single record  
        Employee emp = db.Employees.Where(temp => temp.EmpID ==  
1).FirstOrDefault();  
  
        //read data row-by-row  
        Console.WriteLine("Emp ID: " + emp.EmpID);  
        Console.WriteLine("Emp Name: " + emp.EmpName);  
        Console.WriteLine("Salary: " + emp.Salary);  
  
        Console.ReadKey();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/319.2. Console - Entity Framework - FirstOrDefault/FirstOrDefaultExample/FirstOrDefaultEx... - X
Emp ID: 1
Emp Name: Scott
Salary: 4000
```

Note: If any database connection problem, it shows exception (run time error).

HARSHA

Entity Framework – Insertion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

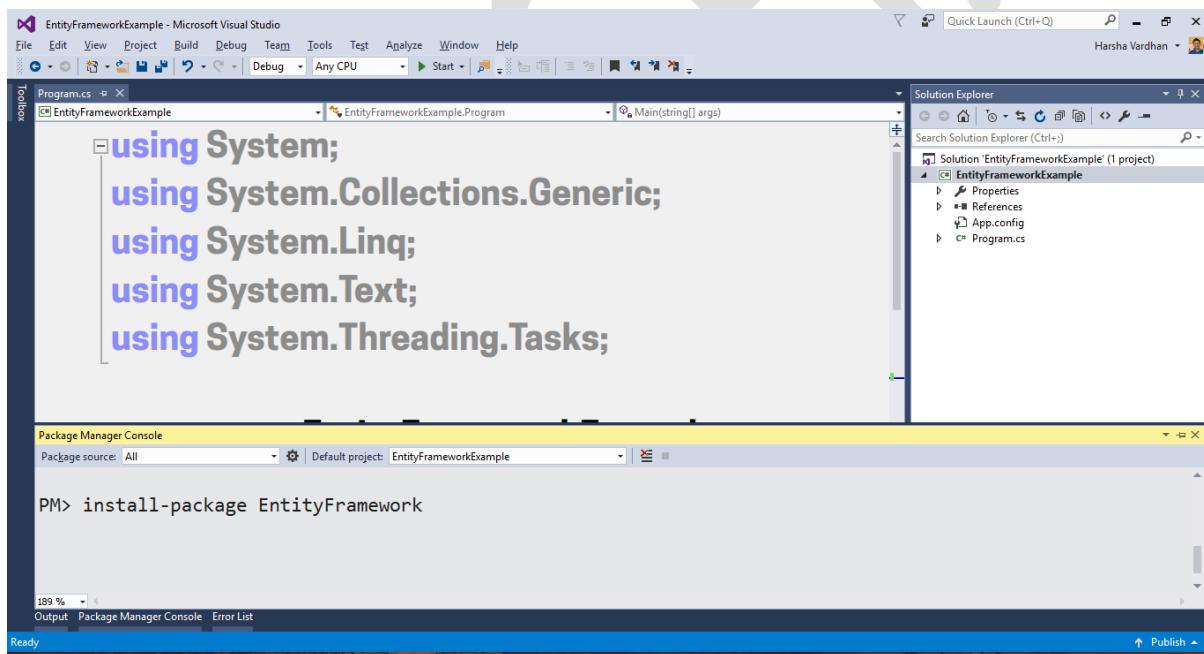
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “EfInsertionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EfInsertionExample”.
- Click on OK.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

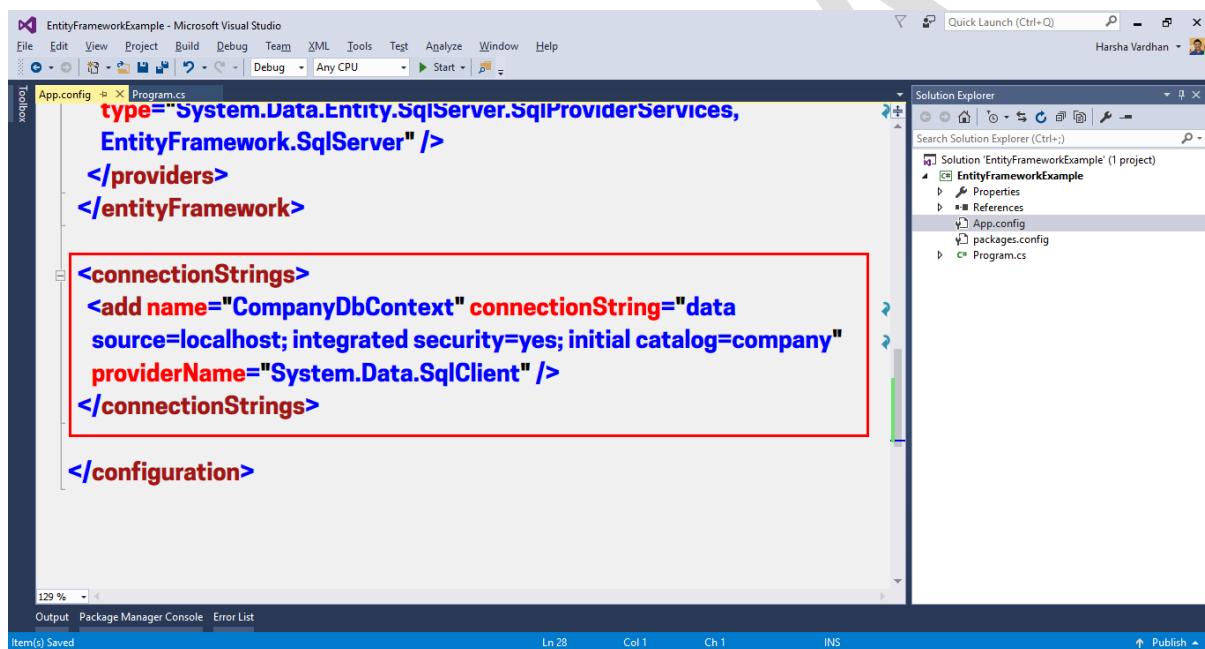
Install-package EntityFramework



Adding “ConnectionString” in “App.Config”:

- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

```
<connectionStrings>
  <add name="CompanyDbContext" connectionString="data
  source=localhost; integrated security=yes; initial
  catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>
```



Program.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Collections.Generic;
using System.Linq;
```

```
namespace EfInsertionExample
{
    public class Employee
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int EmpID { get; set; }
        public string EmpName { get; set; }
        public decimal Salary { get; set; }
    }

    public class CompanyDbContext : DbContext
    {
        public DbSet<Employee> Employees { get; set; }
    }

    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());

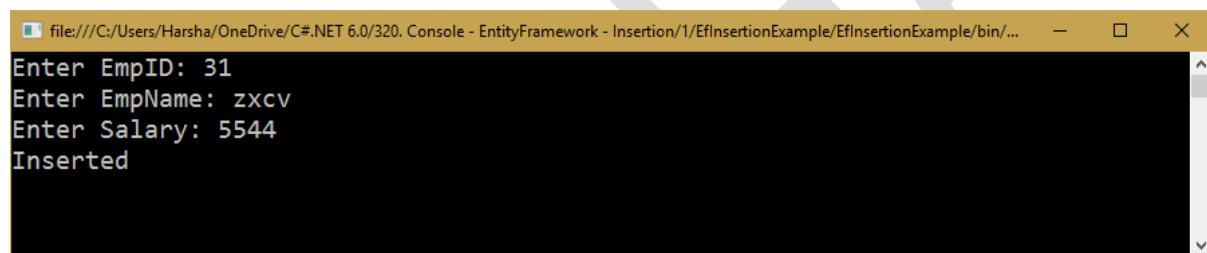
            //insertion
            CompanyDbContext db = new CompanyDbContext();
            Employee emp = new Employee();
            emp.EmpID = empId;
            emp.EmpName = empName;
            emp.Salary = sal;
            db.Employees.Add(emp);
            db.SaveChanges();
        }
    }
}
```

```
        Console.WriteLine("Inserted");
        Console.ReadKey();
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/320. Console - EntityFramework - Insertion/1/EfInsertionExample/EfInsertionExample/bin/... — □ X
Enter EmpID: 31
Enter EmpName: zxcv
Enter Salary: 5544
Inserted
```

Entity Framework – Updation - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

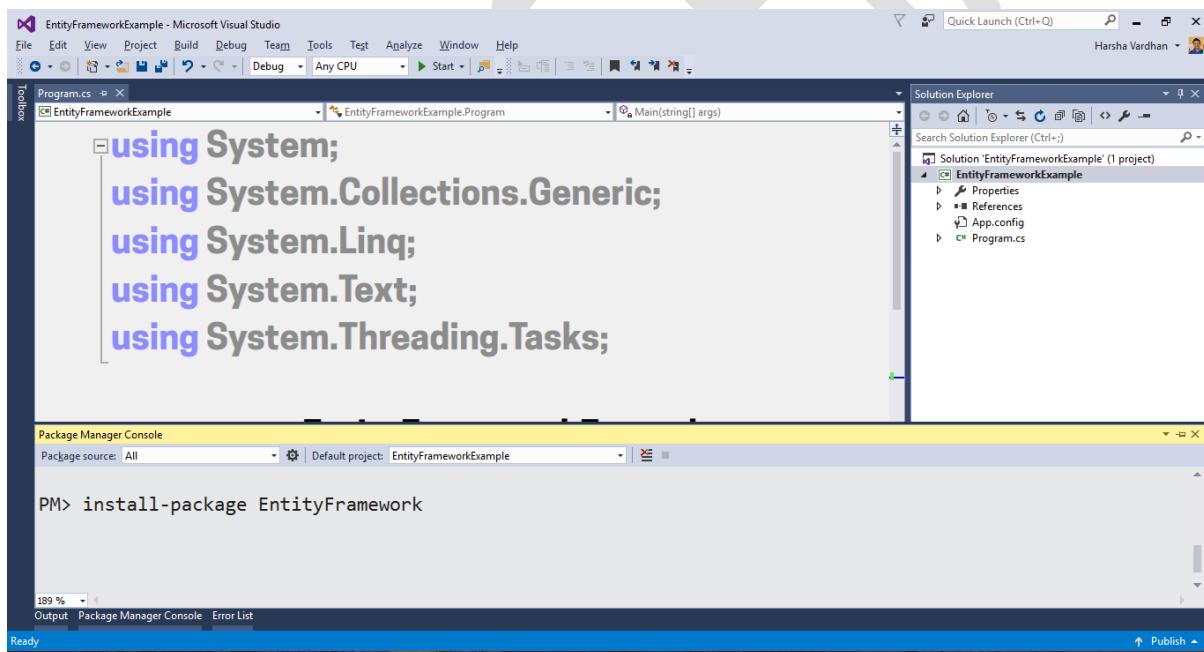
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “EfUpdationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EfUpdationExample”.
- Click on OK.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

Install-package EntityFramework



Adding “ConnectionString” in “App.Config”:

- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

<connectionStrings>

```
<add name="CompanyDbContext" connectionString="data  
source=localhost; integrated security=yes; initial  
catalog=company" providerName="System.Data.SqlClient" />  
</connectionStrings>
```



A screenshot of Microsoft Visual Studio showing the configuration file (App.config) for an Entity Framework example. The connection string section is highlighted with a red box.

```
<configuration>  
  <connectionStrings>  
    <add name="CompanyDbContext" connectionString="data  
      source=localhost; integrated security=yes; initial catalog=company"  
      providerName="System.Data.SqlClient" />  
  </connectionStrings>  
</configuration>
```

The Solution Explorer on the right shows the project structure with files like EntityFrameworkExample.cs, Properties, References, App.config, packages.config, and Program.cs.

Program.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
using System.Collections.Generic;
using System.Linq;

namespace EfUpdationExample
{
    public class Employee
    {
        [Key]
        public int EmpID { get; set; }
        public string EmpName { get; set; }
        public decimal Salary { get; set; }
    }

    public class CompanyDbContext : DbContext
    {
        public DbSet<Employee> Employees { get; set; }
    }

    class Program
    {
        static void Main()
        {
            //accept values from keyboard
            Console.Write("Enter Existing EmpID: ");
            int empId = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter EmpName: ");
            string empName = Console.ReadLine();
            Console.Write("Enter Salary: ");
            decimal sal = Convert.ToDecimal(Console.ReadLine());

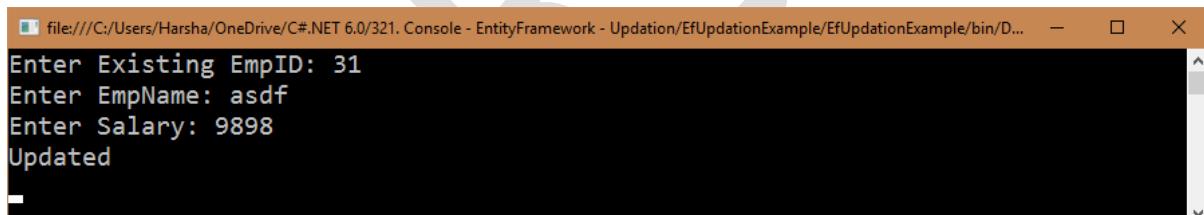
            //updation
            CompanyDbContext db = new CompanyDbContext();
```

```
Employee emp = db.Employees.Where(temp => temp.EmpID ==  
empId).FirstOrDefault();  
emp.EmpName = empName;  
emp.Salary = sal;  
db.SaveChanges();  
  
Console.WriteLine("Updated");  
Console.ReadKey();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```
file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/321. Console - EntityFramework - Updation/EfUpdationExample/EfUpdationExample/bin/D... — X  
Enter Existing EmpID: 31  
Enter EmpName: asdf  
Enter Salary: 9898  
Updated
```

Entity Framework – Deletion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

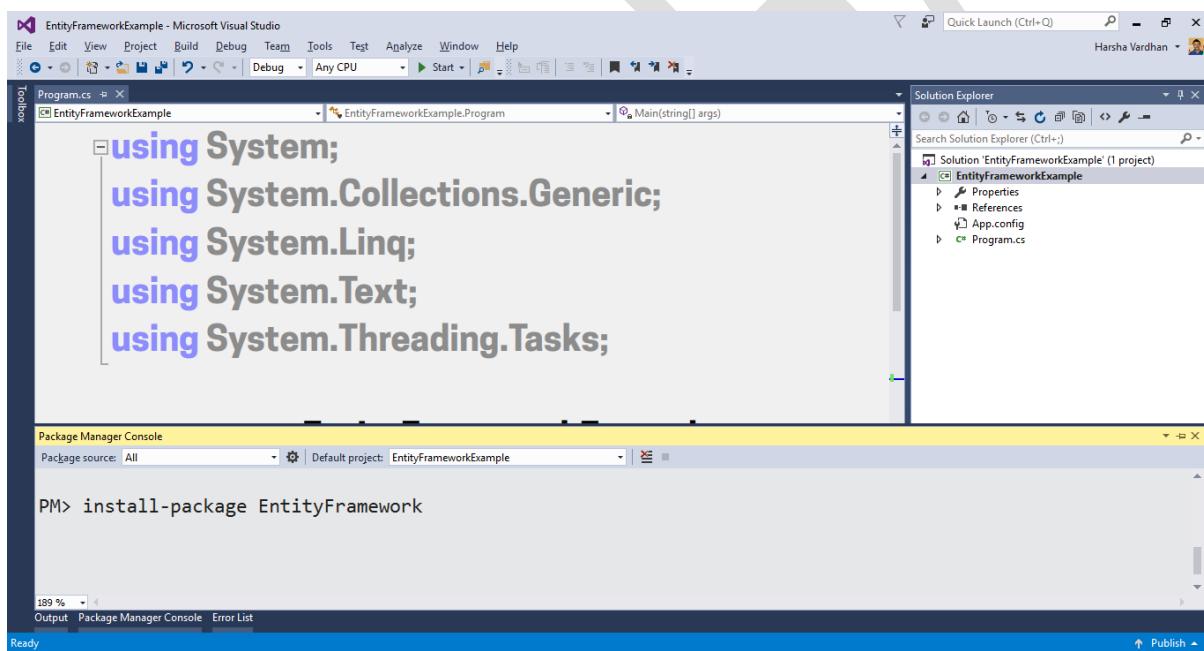
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Console Application”.
- Type the project name as “EfDeletionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EfDeletionExample”.
- Click on OK.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

Install-package EntityFramework



Adding “ConnectionString” in “App.Config”:

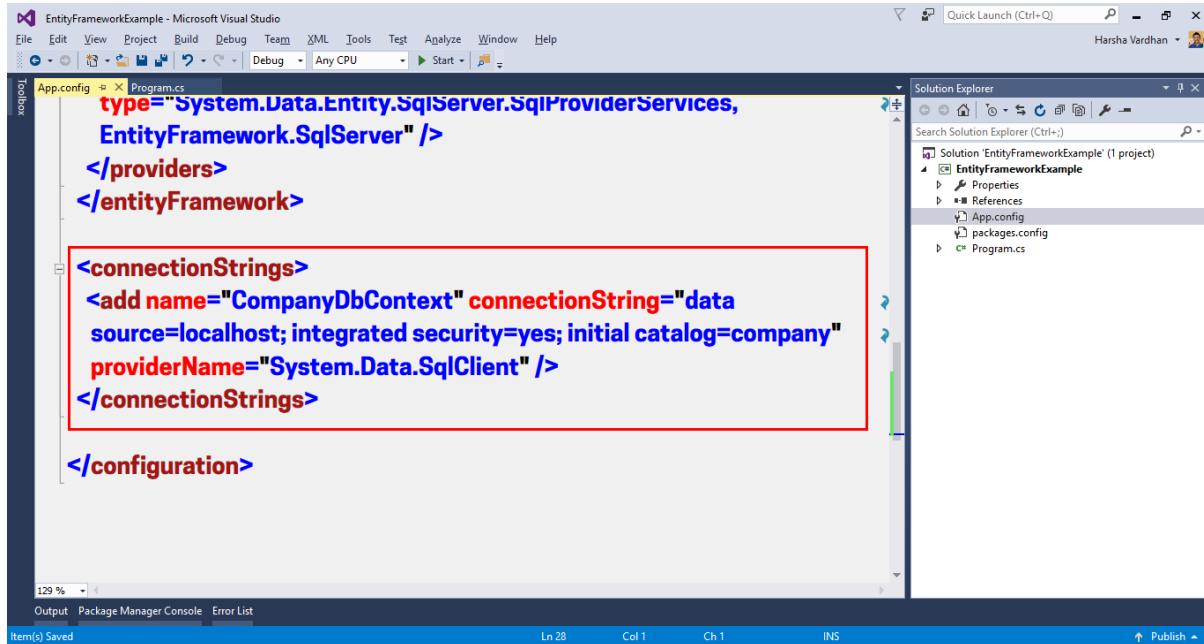
- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

<connectionStrings>

```

<add name="CompanyDbContext" connectionString="data
source=localhost; integrated security=yes; initial
catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>

```



Program.cs

```

using System;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;
using System.Collections.Generic;
using System.Linq;

namespace EfDeletionExample
{
    public class Employee
    {
        [Key]
        public int EmpID { get; set; }
        public string EmpName { get; set; }
        public decimal Salary { get; set; }
    }
}

```

```

public class CompanyDbContext : DbContext
{
    public DbSet<Employee> Employees { get; set; }
}

class Program
{
    static void Main()
    {
        //accept values from keyboard
        Console.Write("Enter Existing EmpID: ");
        int empId = Convert.ToInt32(Console.ReadLine());

        //deletion
        CompanyDbContext db = new CompanyDbContext();
        Employee emp = db.Employees.Where(temp => temp.EmpID == empId).FirstOrDefault();
        db.Employees.Remove(emp);
        db.SaveChanges();

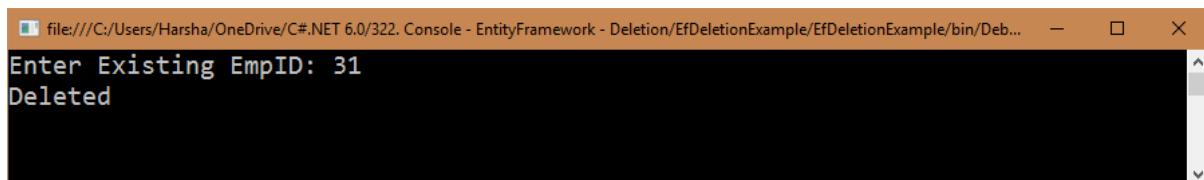
        Console.WriteLine("Deleted");
        Console.ReadKey();
    }
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



```

file:///C:/Users/Harsha/OneDrive/C#.NET 6.0/322. Console - EntityFramework - Deletion/EfDeletionExample/EfDeletionExample/bin/Deb...
Enter Existing EmpID: 31
Deleted

```

C#.NET – Windows Forms Applications

Introduction to Windows Forms Applications

- “Windows forms applications” are “GUI (Graphical User Interface)” based applications that can be developed in .NET.
- Windows forms applications are also called as “WinForms”.
- Windows forms applications are created using c#.net.
- Windows forms applications are user-friendly and easy to use.
- Windows forms applications are created based on Object-oriented programming concepts.
- Windows forms application is a collection of windows forms.
- A windows form (win form) is a GUI container and a window that contains UI (user interface), using which the user can interact with the system. Ex: Login form
- Windows forms applications can interact with databases using ADO.NET.
- Windows forms applications support “local GUI” that runs on windows operating system locally.
- Windows forms applications are compiled into “.exe” file.

Ex: WindowsFormsApplication1.exe

- The exe file is also called as “assembly”.
- The exe file contains compiled source code (in MSIL (Microsoft Intermediate Language)) of windows forms application.

Assemblies (dll files) to develop windows forms applications:

- .NET provides the following assemblies (dll files) to develop windows forms applications:

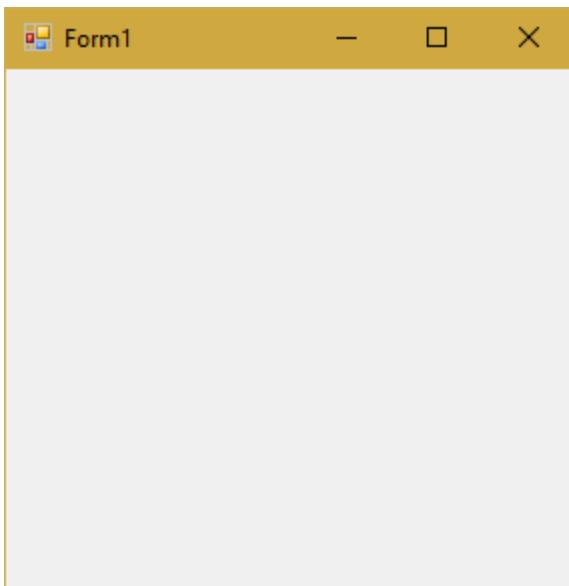
1. System.Windows.Forms.dll
2. System.Drawing.dll

System.Windows.Forms.dll

- “**System.Windows.Forms.dll**” assembly
 - “**System.Windows.Forms**” namespace
 - “Form” class
 - “Label” class
 - “Button” class
 - “TextBox” class
 - etc.

The “System.Windows.Forms.Form” class

- “Form” is a class, which is a member of “System.Windows.Forms” is a namespace.
- The object of “Form” class represents a form.
- Every form contains basic look and feel such as title bar, minimize box, maximize box, close box etc.



Programming Model of Windows Form

- To create a form, you have to create a child class for “System.Windows.Forms.Form” class & you have to create an object for the child class.
- The child class can be “public” class optionally.
- The child class must inherit from “System.Windows.Forms.Form” class. The “System.Windows.Forms.Form” class provides the layout (appearance), properties, events and methods for the child class.

Syntax:

```
public partial class Classname : System.Windows.Forms.Form  
{  
    public Classname()  
    {  
        InitializeComponent();  
    }  
}
```

- The child class must be “partial class”.
 - The visual studio generated designer code (Form1.Designer.cs) and user-defined code (Form1.cs) should be combined as a single class at compile-time. To make this possible, the form class must be “partial class”.
- “public Classname()”
 - It is the constructor that executes as soon as an object for the form class is created at run time.
- InitializeComponent()
 - The InitializeComponent() method initializes the basic look and feel of the form. The definition of InitializeComponent() method is generated in “Form1.Designer.cs” file.
- Program.cs
 - In Program.cs, at main method, the following statement will create an object for the form class.

```
Application.Run(new ChildClassname());
```

Windows Forms Application – First Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “WindowsFormsApplication1”.

- Type the location as “C:\CSharp”.
- Type the solution name as “WindowsFormsApplication1”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

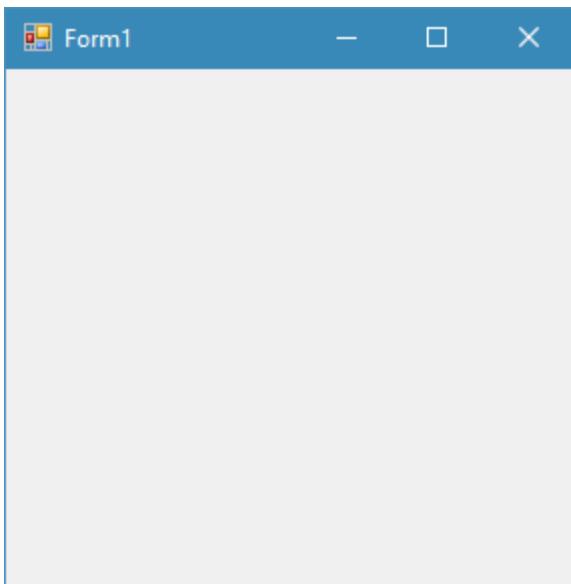
```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Form Constructor - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormConstructorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormConstructorExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;  
using System.Windows.Forms;
```

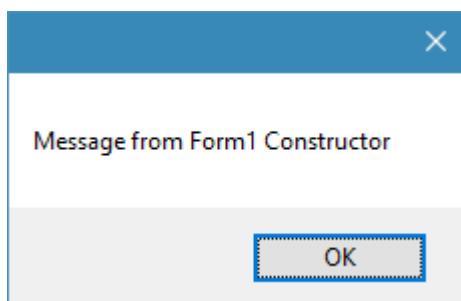
```
namespace FormConstructorExample
```

```
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
            MessageBox.Show("Message from Form1 Constructor");  
        }  
    }  
}
```

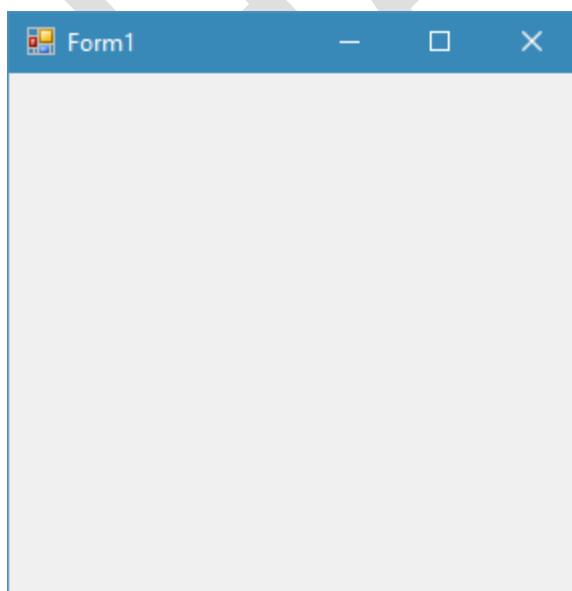
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on OK



Properties of "System.Windows.Forms.Form" class

Properties of "System.Windows.Forms.Form" class

- The properties of "System.Windows.Forms.Form" class provides a set of properties to customize the look and feel of the form.

Sl. No	Property	Description
1	Text	Represents title of the form that appears in the titlebar of the form. <u>Syntax:</u> this.Text = "title here";
2	ShowIcon	Shows / hides the form icon. <u>True:</u> Shows the form icon. <u>False:</u> Hides the form icon. <u>Syntax:</u> this.ShowIcon = true false;
3	ShowInTaskBar	Shows / hides the form icon in the windows taskbar. <u>True:</u> Shows the form icon in the windows taskbar. <u>False:</u> Hides the form icon in the windows taskbar. <u>Syntax:</u> this.ShowInTaskBar = true false;
4	MinimizeBox	Enables / Disables the minimize box. <u>True:</u> Enables the minimize box. <u>False:</u> Disables the minimize box. <u>Syntax:</u> this.MinimizeBox = true false;

5	MaximizeBox	Enables / disables the maximize box. <u>True:</u> Enabled the maximize box. <u>False:</u> Disables the maximize box. <u>Syntax:</u> this.MaximizeBox = true false;
6	ControlBox	Shows / hides the control box (minimize, maximize, close). <u>True:</u> Shows the control box. <u>False:</u> Hides the control box. <u>Syntax:</u> this.ControlBox = true false;
7	TopMost	Determines whether the form should appear always on the top of other windows. <u>True:</u> Shows the form always on top. <u>False:</u> Don't shows the form always on top. <u>Syntax:</u> this.TopMost = true false;
8	WindowState	Represents current status of the form. <u>Options:</u> Normal Minimized Maximized <u>Syntax:</u> this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
9	FormBorderStyle	Enables / disabled the form resizing. <u>Options:</u> Sizable FixedSingle <u>Syntax:</u> this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
10	Cursor	Represents mouse cursor of the form. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> this.Cursor = System.Windows.Forms.Cursors.Hand;

11	BackColor	Represents background color of the form. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> this.BackColor = System.Drawing.Color.Green;
12	BackgroundImage	Represents background image of the form. <u>Options:</u> Any image file. <u>Syntax:</u> this.BackgroundImage = System.Drawing.Image.FromFile("image file path");
13	Size	Represents size (width and height) of the form. <u>Options:</u> pixels. <u>Syntax:</u> this.Size = new System.Drawing.Size(int width, int height);
13	Location	Represents position (X and Y) of the form on the screen. <u>Options:</u> pixels <u>Syntax:</u> this.Location = new System.Drawing.Point(int x, int y);
14	Icon	Represents form icon. <u>Options:</u> Any icon file (.ico) <u>Syntax:</u> this.Icon = new System.Drawing.Icon("icon file path");

“System.Windows.Forms.Form.Text” property - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TextExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “TextExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

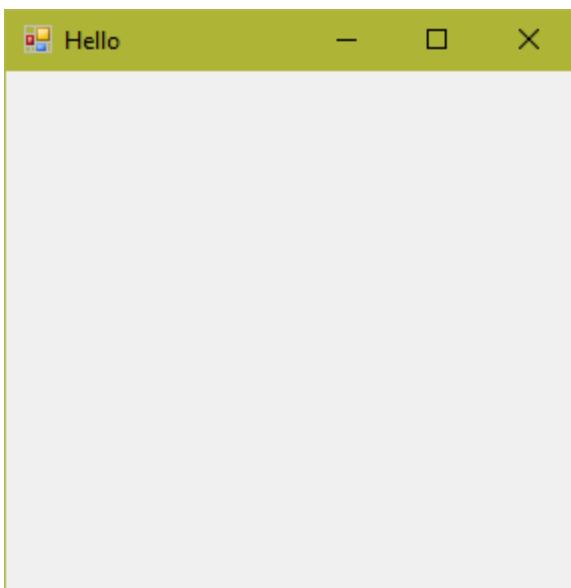
```
using System;
using System.Windows.Forms;

namespace TextExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Text = "Hello";
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the form title is changed as "Hello".

"System.Windows.Forms.Form. ShowIcon" property - Example

Creating Project

- Open Visual Studio 2019. Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8". Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "FormShowIconExample".
- Type the location as "C:\CSharp".
- Type the solution name as "FormShowIconExample".
- Click on OK.
- Right click on the form and click on "View Code".

Form1.cs

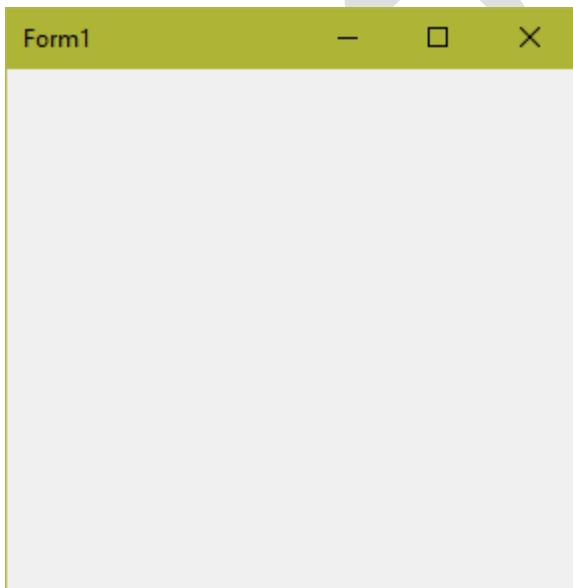
```
using System;
using System.Windows.Forms;
```

```
namespace FormShowIconExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.ShowIcon = false;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the form icon doesn't appear.

“System.Windows.Forms.Form. ShowInTaskBar” property -Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormShowInTaskBarExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormShowInTaskBarExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

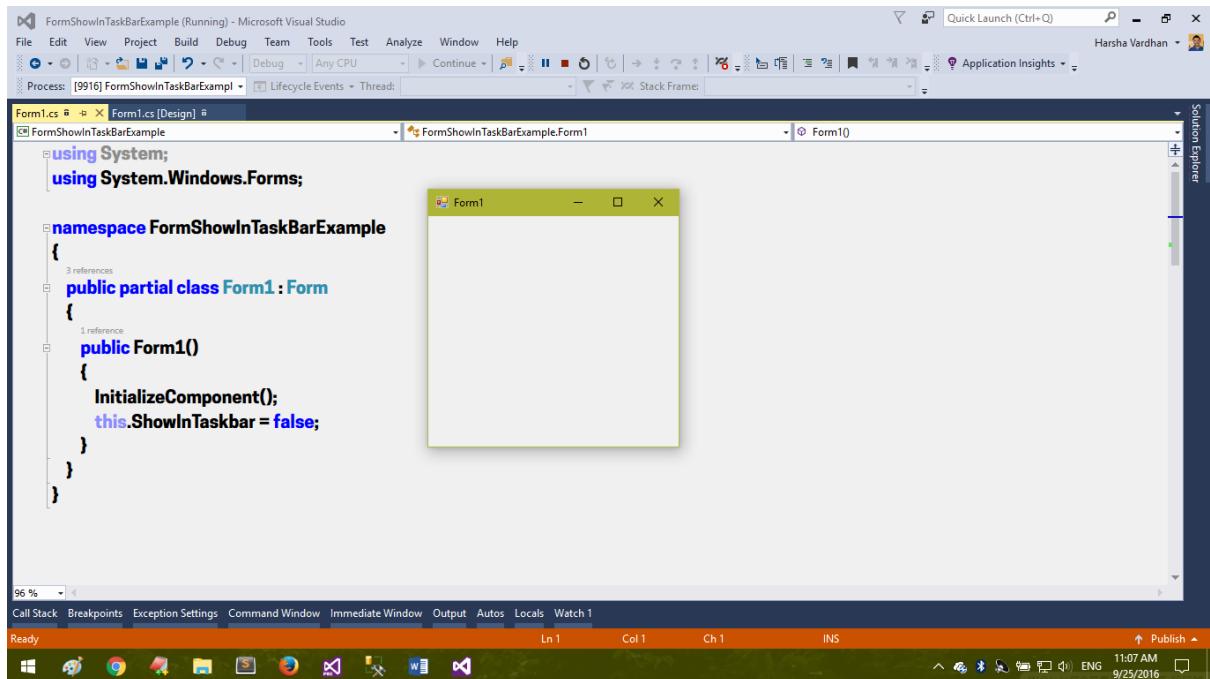
Form1.cs

```
using System;
using System.Windows.Forms;

namespace FormShowInTaskBarExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.ShowInTaskbar = false;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Note: You can notice that the form icon doesn't appear in the windows taskbar.

“System.Windows.Forms.Form.MinimizeBox” property - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormMinimizeBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormMinimizeBoxExample”.

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

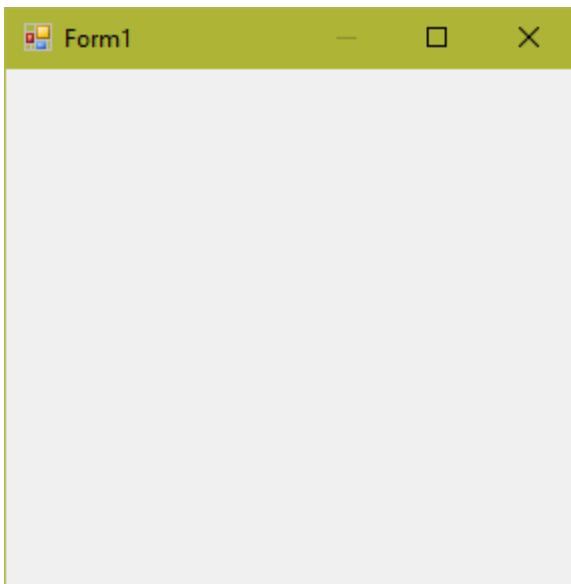
```
using System;
using System.Windows.Forms;

namespace FormMinimizeBoxExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.MinimizeBox = false;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the minimize box doesn't work.

"System.Windows.Forms.Form. MaximizeBox" property - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "FormMaximizeBoxExample".
- Type the location as "C:\CSharp".
- Type the solution name as "FormMaximizeBoxExample".
- Click on OK.
- Right click on the form and click on "View Code".

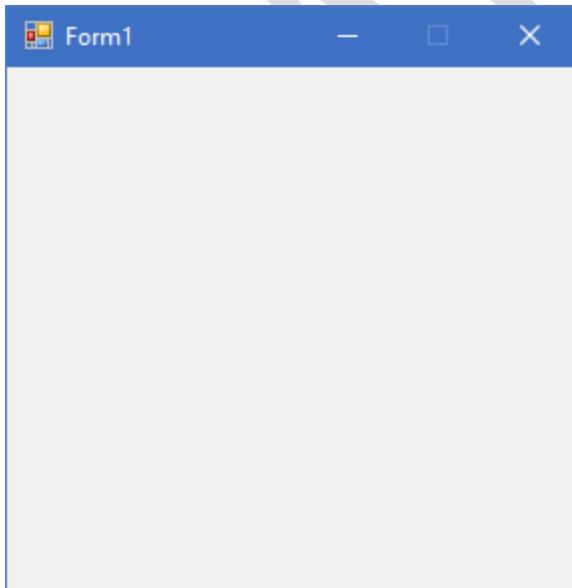
Form1.cs

```
using System;
using System.Windows.Forms;

namespace FormMaximizeBoxExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.MaximizeBox = false;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Now the maximize box doesn't work.

“System.Windows.Forms.Form. ControlBox” property

- Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormControlBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormControlBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

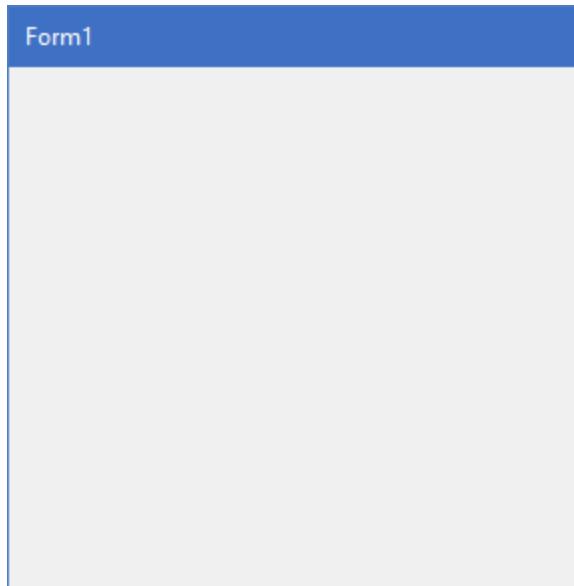
Form1.cs

```
using System;
using System.Windows.Forms;

namespace FormControlBoxExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.ControlBox = false;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Now the control box doesn't appear.

"System.Windows.Forms.Form. TopMost" property - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "FormTopMostExample".
- Type the location as "C:\CSharp".
- Type the solution name as "FormTopMostExample".
- Click on OK.
- Right click on the form and click on "View Code".

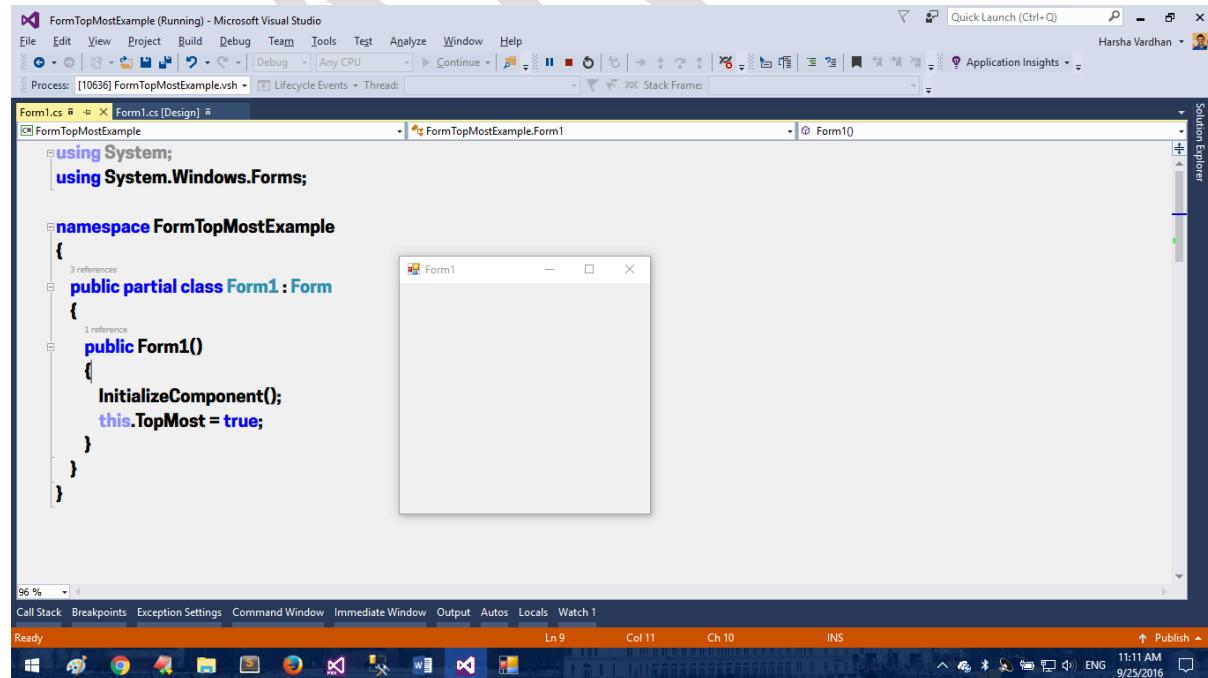
Form1.cs

```
using System;
using System.Windows.Forms;

namespace FormTopMostExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.TopMost = true;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Now the form is always on the top of other windows.

“System.Windows.Forms.Form.WindowState” property –Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormWindowStateExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormWindowStateExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

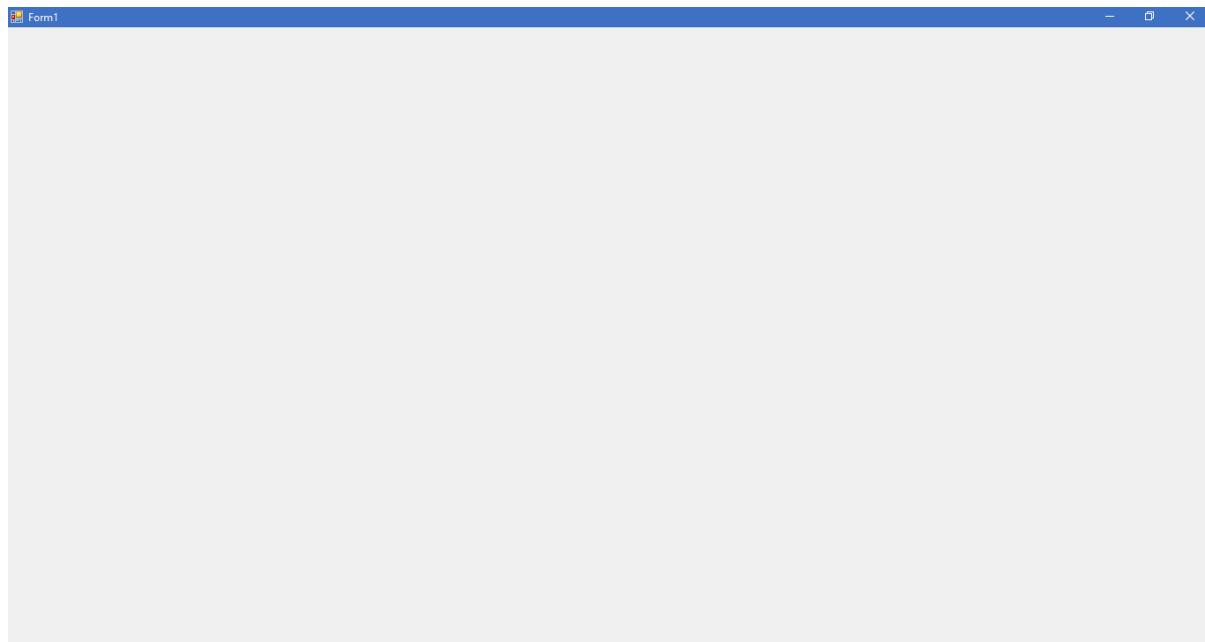
```
using System;
using System.Windows.Forms;

namespace FormWindowStateExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.WindowState = FormWindowState.Maximized;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the form is automatically maximized.

“System.Windows.Forms.Form. FormBorderStyle” property – Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormBorderStyleExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “FormBorderStyleExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

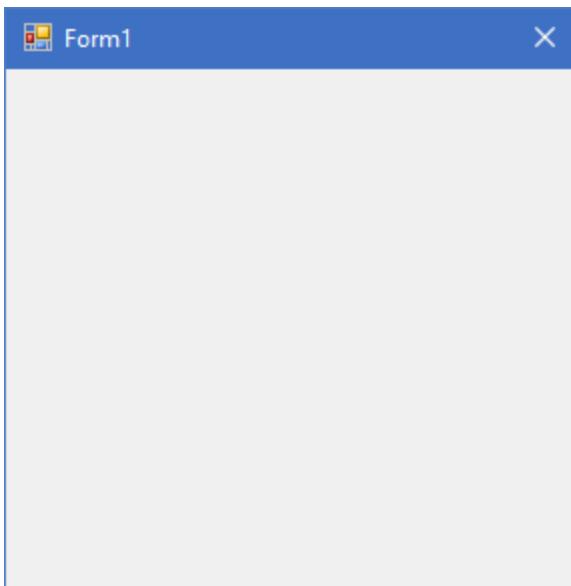
```
using System;
using System.Windows.Forms;

namespace FormBorderStyleExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.MinimizeBox = false;
            this.MaximizeBox = false;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the form isn't resizable.

"System.Windows.Forms.Form.Cursor" property - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "FormCursorExample".
- Type the location as "C:\CSharp".
- Type the solution name as "FormCursorExample". Click on OK.
- Right click on the form and click on "View Code".

Form1.cs

```
using System;
```

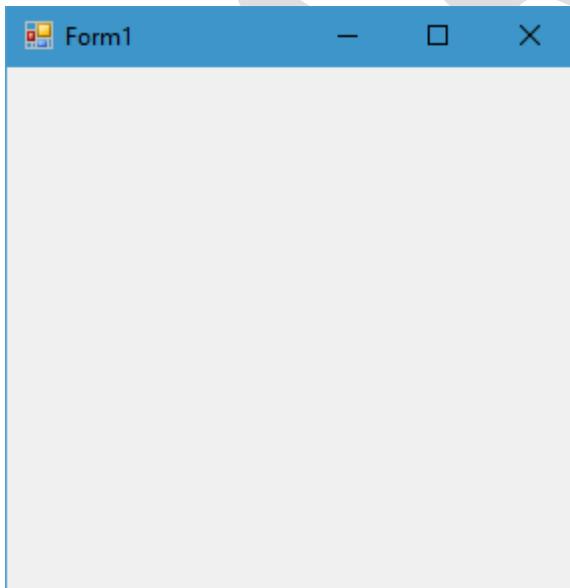
```
using System.Windows.Forms;
```

```
namespace FormCursorExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Cursor = Cursors.Hand;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the mouse pointer is changed as hand symbol.

“System.Windows.Forms.Form. BackColor” property - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormBackColorExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormBackColorExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

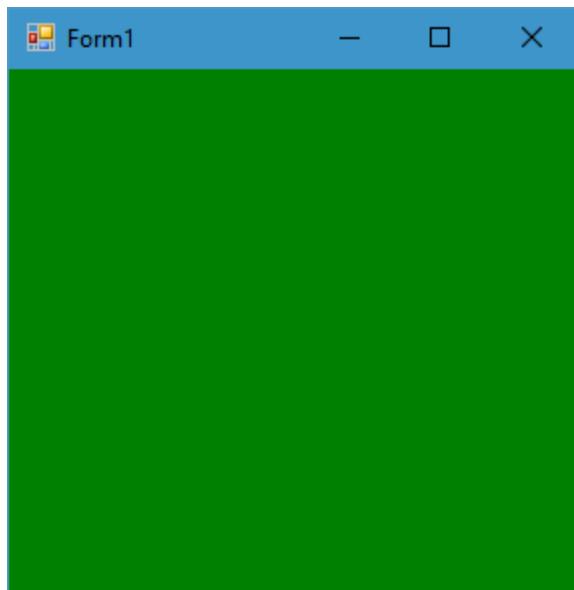
Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FormBackColorExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.BackColor = Color.Green;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

The form has background color.

"System.Windows.Forms.Form. BackgroundImage" property – Example

Creating Project

- Copy and paste "img1.jpg" into "C:\CSharp" folder.
- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "FormBackgroundImageExample".
- Type the location as "C:\CSharp".
- Type the solution name as "FormBackgroundImageExample".
- Click on OK.

- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FormBackgroundImageExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            string s = @"C:\CSharp\img1.jpg";
            this.BackgroundImage = Image.FromFile(s);
            this.BackgroundImageLayout = ImageLayout.Stretch;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The form has background image.

"System.Windows.Forms.Form. Size" property - Example

Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "FormSizeExample".
- Type the location as "C:\CSharp".
- Type the solution name as "FormSizeExample". Click on OK.
- Right click on the form and click on "View Code".

Form1.cs

```
using System;
```

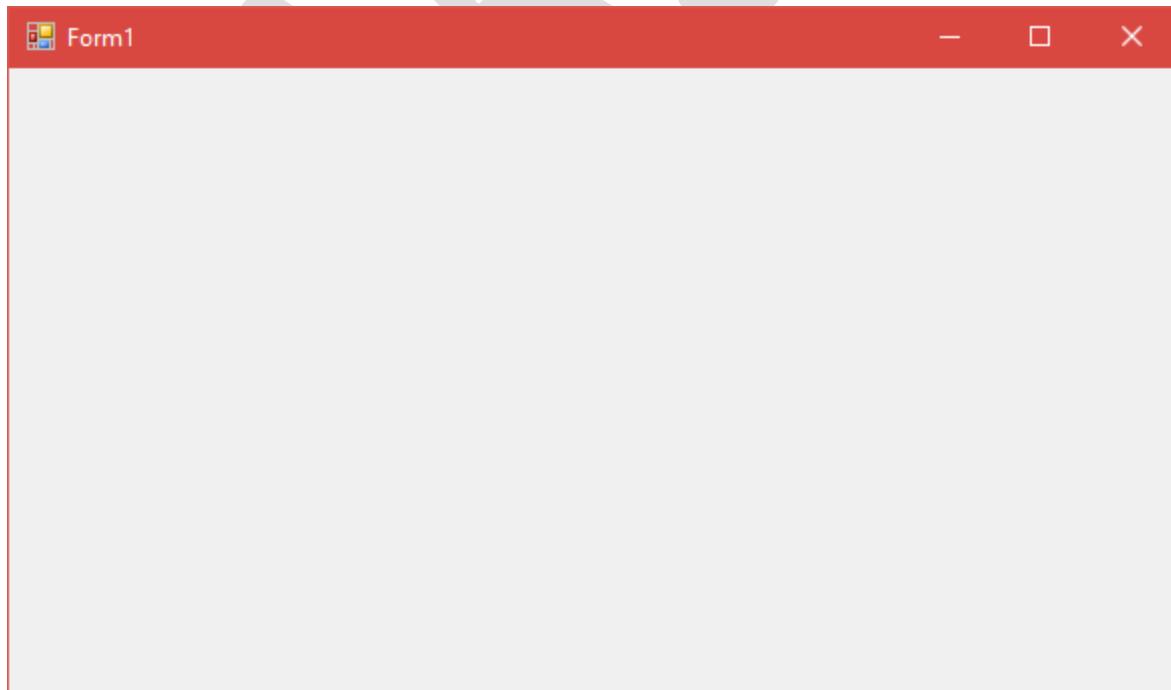
```
using System.Drawing;
using System.Windows.Forms;

namespace FormSizeExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Size = new Size(600, 350);
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The form size has been changed.

“System.Windows.Forms.Form. Location” property - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormLocationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormLocationExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

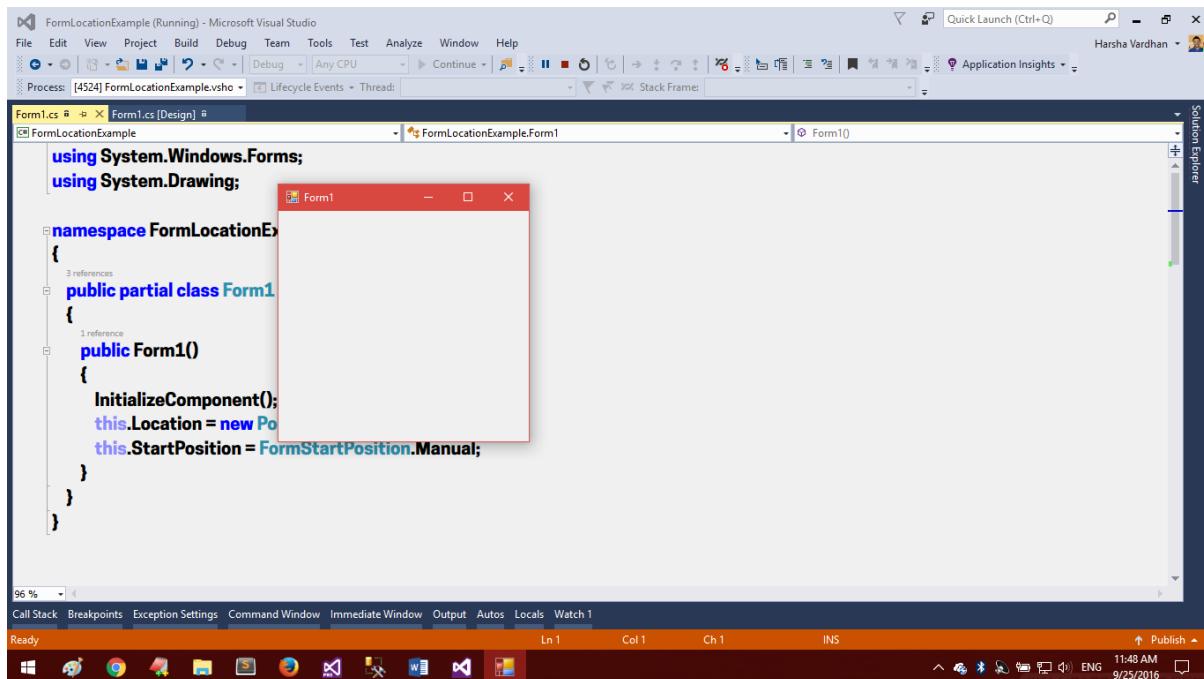
namespace FormLocationExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Location = new Point(300, 200);
            this.StartPosition = FormStartPosition.Manual;
        }
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The form appears exactly at “300 by 200” position on the screen.

“System.Windows.Forms.Form.Icon” property - Example

Creating Project

- Copy and paste “Sample.ico” file into “C:\CSharp” folder.
- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Windows Forms Application”.
- Type the project name as “FormIconExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormIconExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

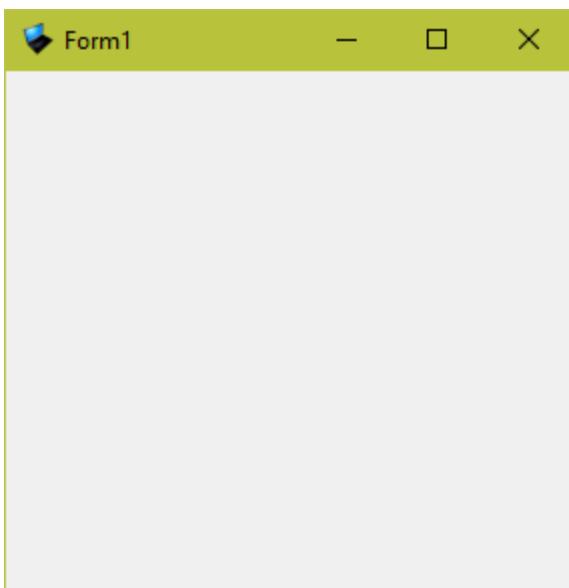
```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FormIconExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Icon = new Icon(@"C:\CSharp\Sample.ico");
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Now the form icon has been changed.

Events of “System.Windows.Forms.Form” class

Events of “System.Windows.Forms.Form” class

- An event represents “action that is performed by the user at run time”.
- An event can call a method (function).
- Events are used to make the form to display output based on the input.

Sl. No	Event	Description
1	Load	<p>Executes after loading the form object into memory at run time.</p> <p>Executes after immediately after form constructor.</p> <p><u>Syntax:</u> <code>this.Load += Form1_Load;</code></p>
2	Shown	<p>Executes after displaying the form on the screen.</p> <p><u>Syntax:</u> <code>thisShown += Form1_Shown;</code></p>

3	Click	Executes when the user clicks on the form. <u>Syntax:</u> this.Click += Form1_Click;
4	DoubleClick	Executes when the user double clicks on the form. <u>Syntax:</u> this.DoubleClick += Form1_DoubleClick;
5	MouseClick	Same as “Click” event; but we can identify the mouse pointer position also in this event. <u>Syntax:</u> this.MouseClick += Form1_MouseClick;
6	MouseMove	Executes when the user moves the mouse pointer from one place to another place within the same form. <u>Syntax:</u> this.MouseMove += Form1_MouseMove;
6	KeyPress	Executes when the user presses any key on the keyboard. We can identify the currently pressed key. <u>Syntax:</u> this.KeyPress += Form1_KeyPress;
7	FormClosing	Executes when the user clicks on the “close” button of the form. We can cancel the form closing. <u>Syntax:</u> this.FormClosing += Form1_FormClosing;

“System.Windows.Forms.Form. Load” event - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormLoadExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormLoadExample”.

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

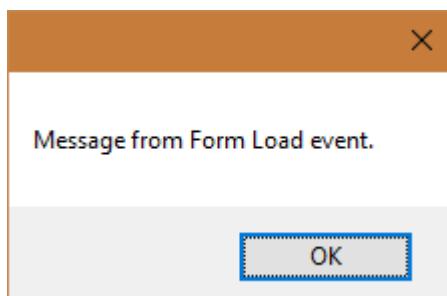
namespace FormLoadExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Load += Form1_Load;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            MessageBox.Show("Message from Form Load event.");
        }
    }
}
```

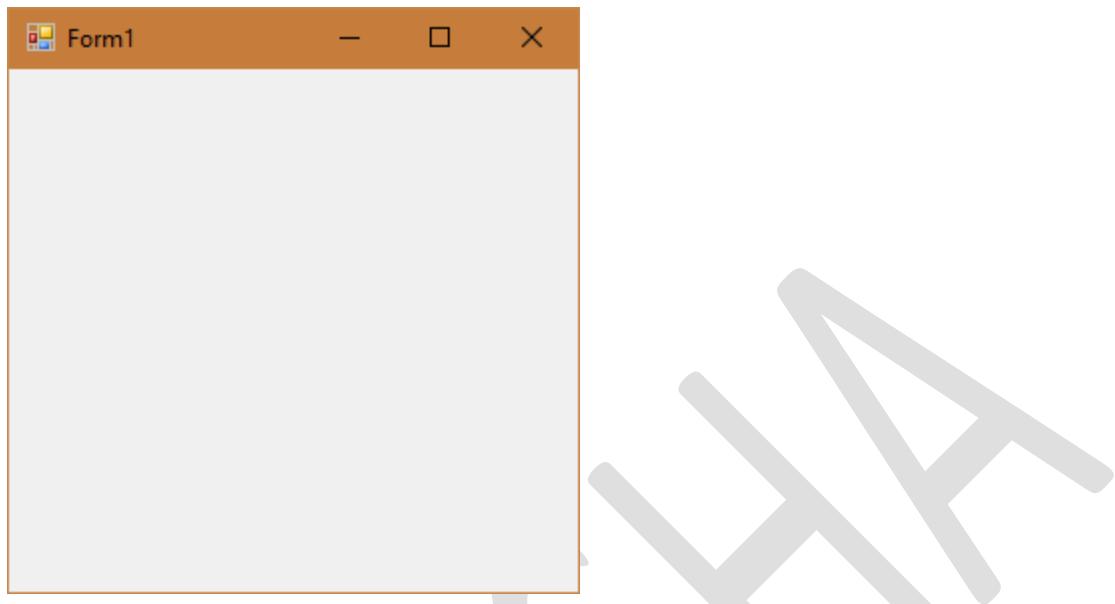
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



First message box appears and then the form appears.



HARSHA

“System.Windows.Forms.Form. Shown” event - Example

Creating Project

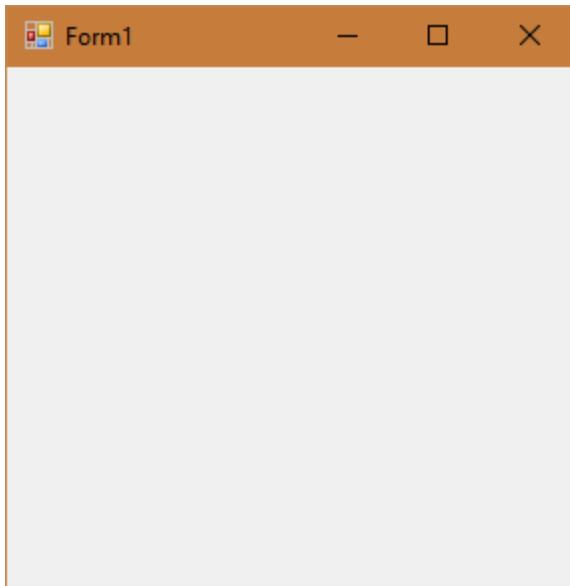
- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormShownExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormShownExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

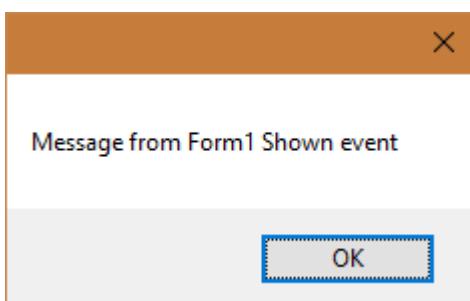
```
using System;
using System.Windows.Forms;
namespace FormShownExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            thisShown += Form1_Shown;
        }
        private void Form1_Shown(object sender, EventArgs e)
        {
            MessageBox.Show("Message from Form1 Shown event");
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

First the form appears and then the message box appears.



“System.Windows.Forms.Form.Click” event - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormClickExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “FormClickExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

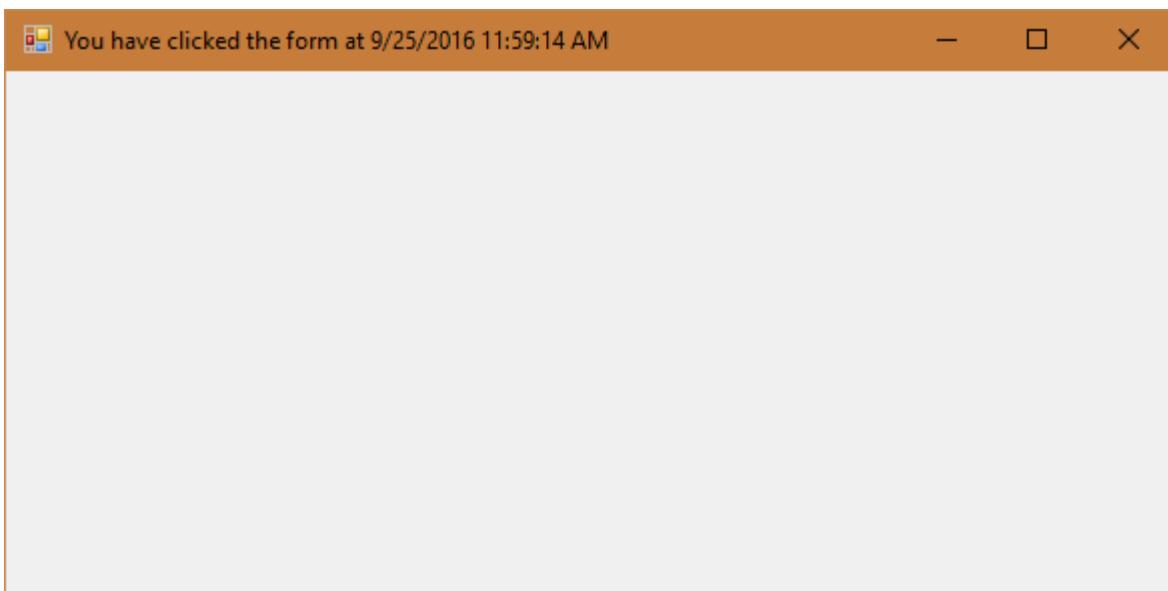
namespace FormClickExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Size = new System.Drawing.Size(600, 300);
            this.Click += Form1_Click;
        }

        private void Form1_Click(object sender, EventArgs e)
        {
            DateTime dt = DateTime.Now;
            string msg = "You have clicked the form at " + dt.ToString();
            this.Text = msg;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When the user clicks on the form, it shows the “You have clicked the form at ...” message, in the form title bar.

“System.Windows.Forms.Form.DoubleClick” event - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DoubleClickExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DoubleClickExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

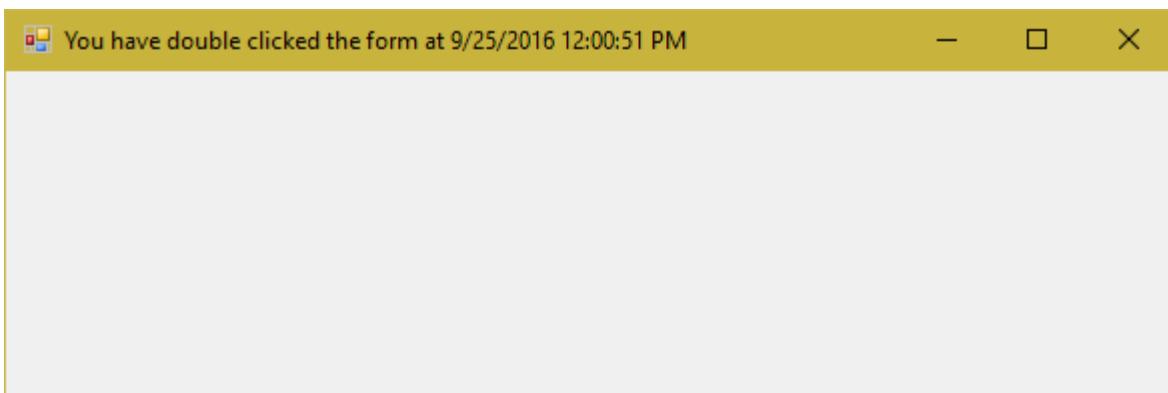
namespace FormDoubleClickExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Size = new System.Drawing.Size(600, 200);
            this.DoubleClick += Form1_DoubleClick;
        }

        private void Form1_DoubleClick(object sender, EventArgs e)
        {
            DateTime dt = DateTime.Now;
            string msg = "You have double clicked the form at " + dt.ToString();
            this.Text = msg;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When the user double clicks on the form, it shows “You have double clicked the form at ...” message, in the form title bar.

“System.Windows.Forms.Form.MouseClick” event - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormMouseClickExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormMouseClickExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

using System;

```
using System.Windows.Forms;
```

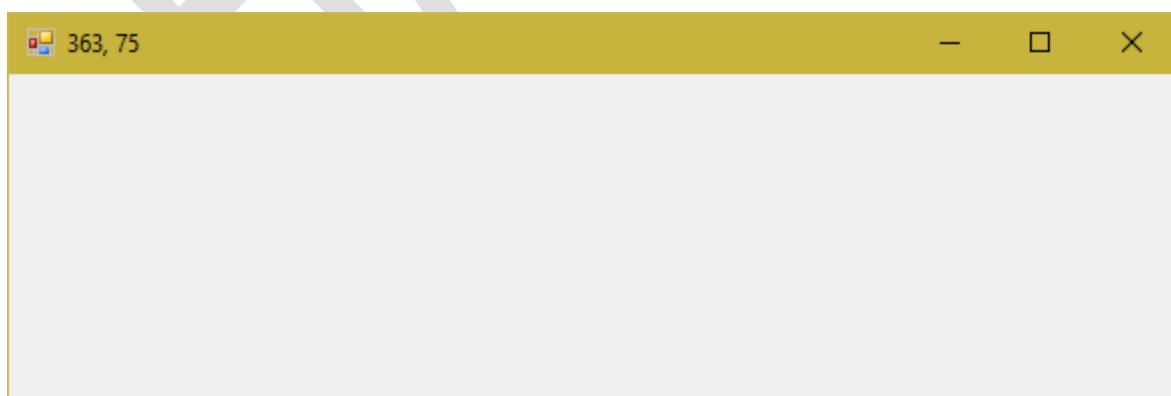
```
namespace FormMouseClickExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Size = new System.Drawing.Size(600, 200);
            this.MouseClick += Form1_MouseClick;
        }

        private void Form1_MouseClick(object sender, MouseEventArgs e)
        {
            string msg = e.X + ", " + e.Y;
            this.Text = msg;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When the user clicks on the form, it shows the X and Y co-ordinates of the mouse pointer.

“System.Windows.Forms.Form. MouseMove” event - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormMouseMoveExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormMouseMoveExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace FormMouseMoveExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.MouseEventHandler += Form1_MouseMove;
        }

        private void Form1_MouseMove(object sender, MouseEventArgs e)
```

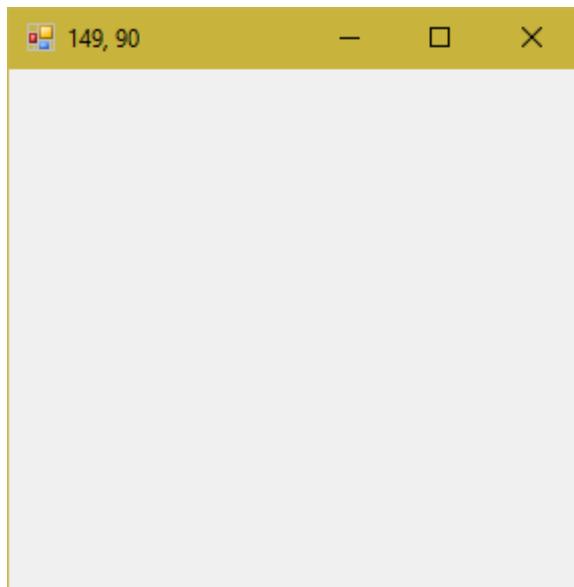
```

{
    int x = e.X;
    int y = e.Y;
    this.Text = x + "," + y;
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

When the user moves the mouse pointer across the form, it shows the X and Y co-ordinates of the mouse pointer.

“System.Windows.Forms.Form. KeyPress” event - Example

Creating Project

- Open Visual Studio 2019.

- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormKeyPressExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormKeyPressExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

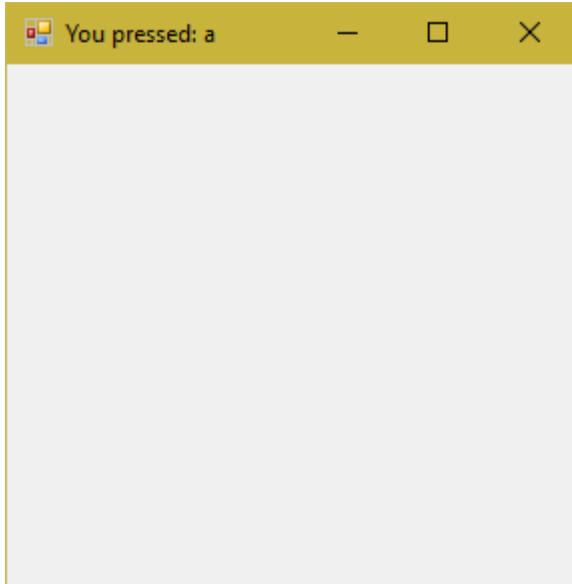
namespace FormKeyPressExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.KeyPress += Form1_KeyPress;
        }

        private void Form1_KeyPress(object sender, KeyPressEventArgs e)
        {
            char ch = e.KeyChar;
            string msg = "You pressed: " + ch;
            this.Text = msg;
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When the user presses any character in the keyboard, it shows the same in the form title bar.

“System.Windows.Forms.Form. FormClosing” event - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormClosingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormClosingExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

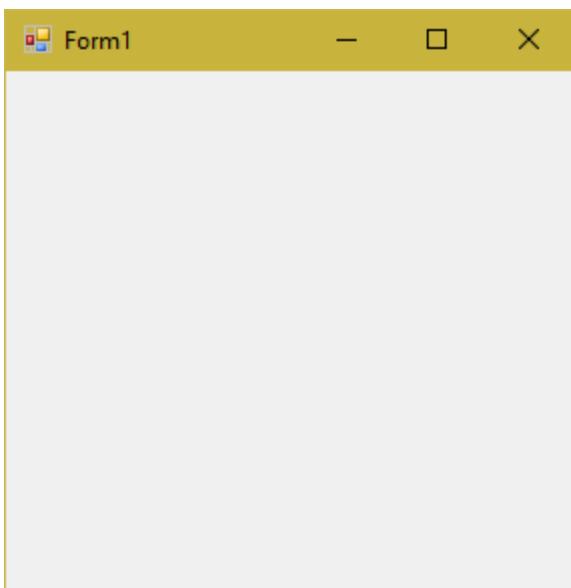
namespace FormClosingExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.FormClosing += Form1_FormClosing;
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            MessageBox.Show("Form is being closed.");
        }
    }
}
```

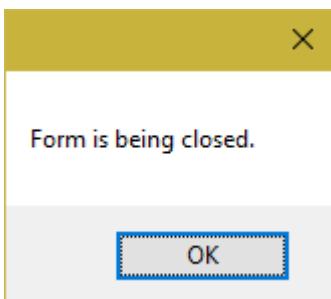
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you click on “Close (X)” button of the form, it shows the following message:



“System.Windows.Forms.Form. FormClosing” event - Example 2

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormClosingExample2”.

- Type the location as “C:\CSharp”.
- Type the solution name as “FormClosingExample2”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace FormClosingExample2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.FormClosing += Form1_FormClosing;
        }

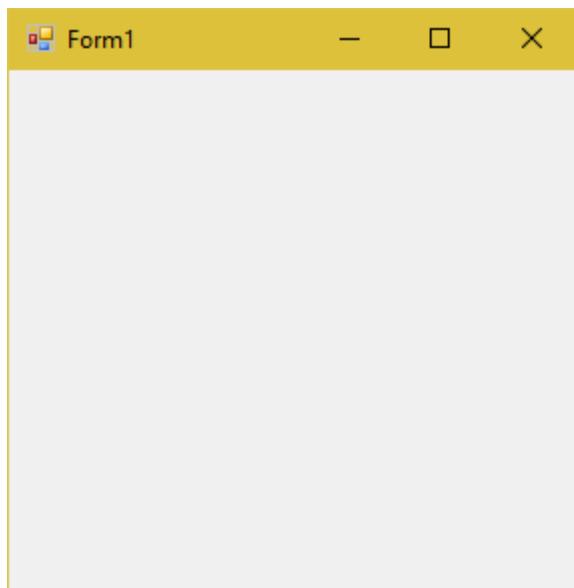
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            DialogResult r = MessageBox.Show("Are you sure to close", "C#.NET",
                MessageBoxButtons.YesNo);

            if (r == DialogResult.Yes)
                e.Cancel = false;
            else
                e.Cancel = true;
        }
    }
}
```

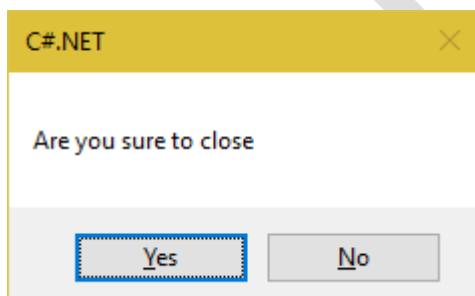
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you click on “Close (X)” button of the form, it shows the confirmation message as follows:



If you click on “Yes”, the form will be closed.

If you click on “No”, the form will not be closed.

Methods of “System.Windows.Forms.Form” class

Methods of “System.Windows.Forms.Form” class

- A method performs some operation on the object.

- The methods of “System.Windows.Forms.Form” class performs some operation based on the form object.

Sl. No	Methods	Description
1	Hide()	Hides the form. <u>Syntax:</u> this.Hide();
2	Show()	Shows the form. <u>Syntax:</u> this.Show();
3	Close()	Closes the form. <u>Syntax:</u> this.Close();

“System.Windows.Forms.Form. Hide” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormHideExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormHideExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

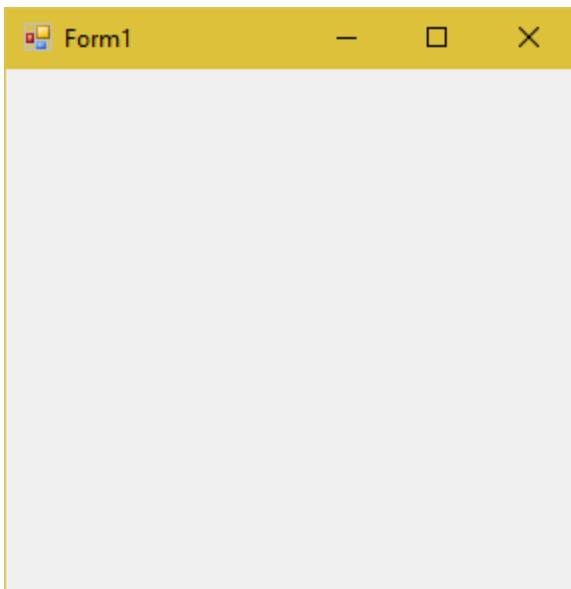
namespace FormHideExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Click += Form1_Click;
        }

        private void Form1_Click(object sender, EventArgs e)
        {
            this.Hide();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you click on the form, the form will be automatically hidden.

Click on “Debug” menu – “Stop Debugging” to stop the application.

“System.Windows.Forms.Form .Show” method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormShowExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormShowExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

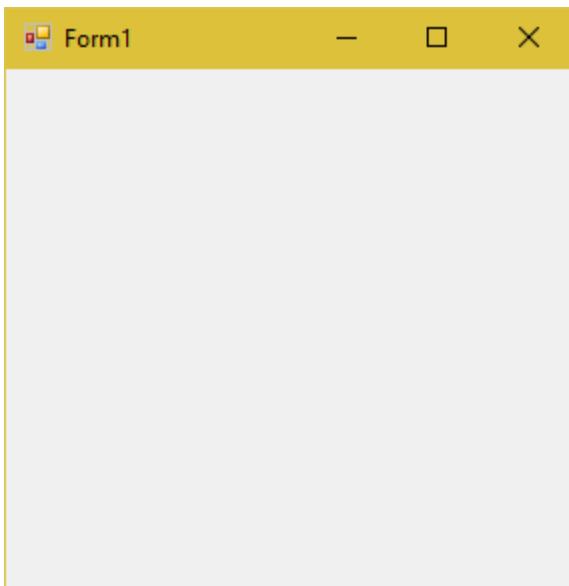
namespace FormShowExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Click += Form1_Click;
        }

        private void Form1_Click(object sender, EventArgs e)
        {
            this.Hide();
            System.Threading.Thread.Sleep(3000); //delay 3 seconds
            this.Show();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you click on the form, the form will be automatically hidden, it automatically appears again in 3 seconds.

"System.Windows.Forms.Form.Close" method - Example

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FormCloseExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FormCloseExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

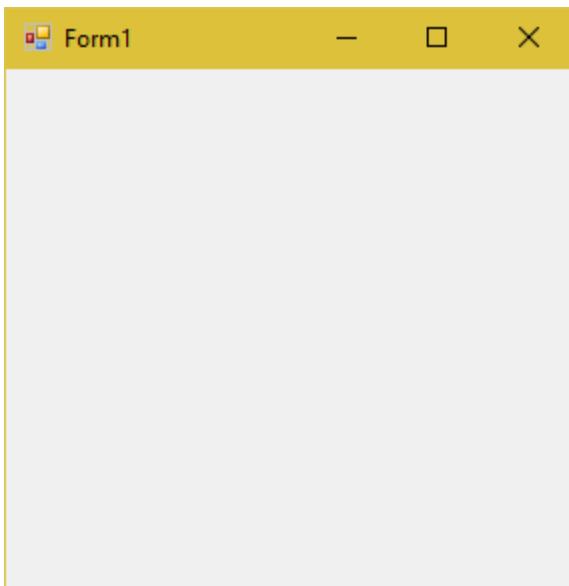
namespace FormCloseExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.Click += Form1_Click;
        }

        private void Form1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you click on the form, the form will be closed.

Introduction to Windows Forms Controls

Introduction to "Windows Forms Controls"

- In windows applications, we use “windows forms controls” to take input from the user and provide output to the user.
- Windows operating system provides a set of controls such as Label, Button, TextBox etc.
- .NET provides a set of pre-defined classes to create controls in a .net windows forms application. Each class represent a control. All those classes are called as “Control Classes”. All the control classes are present in “System.Windows.Forms” namespace.

Control Classes in “System.Windows.Forms” namespace:

- “System.Windows.Forms.dll” assembly:
 - “System.Windows.Forms” namespace:
 1. “Label” class
 2. “Button” class

3. “TextBox” class
4. “NumericUpDown” class
5. “DateTimePicker” class
6. “MonthCalendar” class
7. “MaskedTextBox” class
8. “ErrorProvider” class
9. “CheckBox” class
10. “RadioButton” class
11. “ComboBox” class
12. “ListBox” class
13. “CheckedListBox” class
14. “TreeView” class
15. “PictureBox” class
16. “Panel” class
17. “GroupBox” class
18. “SplitContainer” class
19. “TabControl” class
20. “FlowLayoutPanel” class
21. “LinkLabel” class
22. “Tooltip” class
23. “WebBrowser” class
24. “ProgressBar” class
25. “Timer” class
26. “NotifyIcon” class

27. “ColorDialog” class
28. “FontDialog” class
29. “FolderBrowserDialog” class
30. “OpenFileDialog” class
31. “SaveFileDialog” class
32. “RichTextBox” class
33. “MenuStrip” class
34. “ContextMenuStrip” class
35. “ToolStrip” class
36. “StatusStrip” class
37. “DataGridView” class

“System.Windows.Forms.Label” class

The “System.Windows.Forms.Label” class

- “Label” is a class; “System.Windows.Forms” is a namespace.
- The “Label” class / control is used to display a message in the windows form.

Steps for development of label:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `Label referencevariable;`
- Create an object:

- *referencevariable = new Label();*

- Set properties:
 - *referencevariable.property = value;*

- Add event:
 - *referencevariable.event += method;*

- Add control to the form:
 - *this.Controls.Add(referencevariable);*

Properties of "Label" class:

Sl. No	Property	Description
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <i>referencevariable.Text = "text here";</i></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <i>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</i></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <i>referencevariable.Size = new System.Drawing.Size(int width, int height);</i></p>

4	AutoSize	<p><u>True:</u> The control will take the essential size automatically.</p> <p><u>False:</u> The control will not take the size automatically; we need to set the size manually.</p> <p><u>Syntax:</u> <code>referencevariable.AutoSize = true false;</code></p>
5	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
6	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
7	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
8	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
9	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

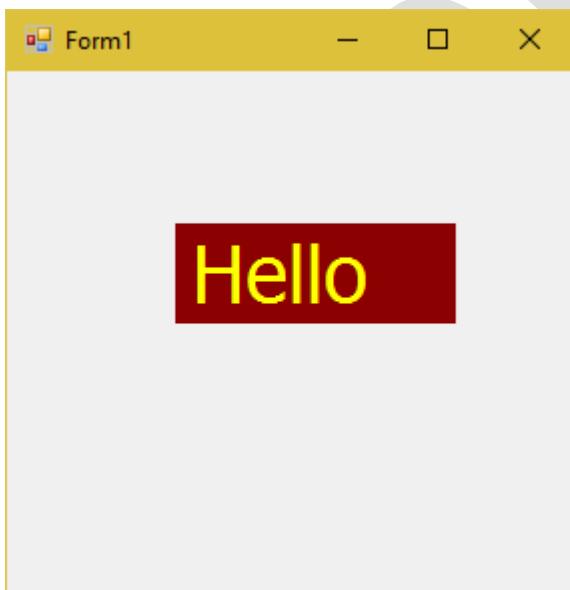
Events of "Label" class:

Sl. No	Event	Description

1	Click	Executes when the user clicks on the control. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>
2	MouseEnter	Executes when the user moves the mouse pointer from outside to inside the control. <u>Syntax:</u> <code>referencevariable.MouseEnter += methodname;</code>
3	MouseLeave	Executes when the user moves the mouse pointer from inside to outside the control. <u>Syntax:</u> <code>referencevariable.MouseLeave += methodname;</code>

“System.Windows.Forms.Label” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.

- Type the project name as “LabelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LabelExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace LabelExample
{
    public partial class Form1 : Form
    {
        //create a reference variable
        Label lbl;

        public Form1()
        {
            InitializeComponent();

            //create an object
            lbl = new Label();

            //set properties
            lbl.Text = "Hello";
            lbl.Font = new System.Drawing.Font("Tahoma", 30);
            lbl.BackColor = System.Drawing.Color.DarkRed;
            lbl.ForeColor = System.Drawing.Color.Yellow;
            lbl.Size = new System.Drawing.Size(140, 50);
            lbl.Location = new System.Drawing.Point(84, 76);
            lbl.Cursor = Cursors.Hand;

            //add event
        }
    }
}
```

```
lbl.Click += Lbl_Click;
lbl.MouseEnter += Lbl_MouseEnter;
lbl.MouseLeave += Lbl_MouseLeave;

//add label to the form
this.Controls.Add(lbl);
}

private void Lbl_Click(object sender, EventArgs e)
{
    lbl.Text = "Thanx";
}

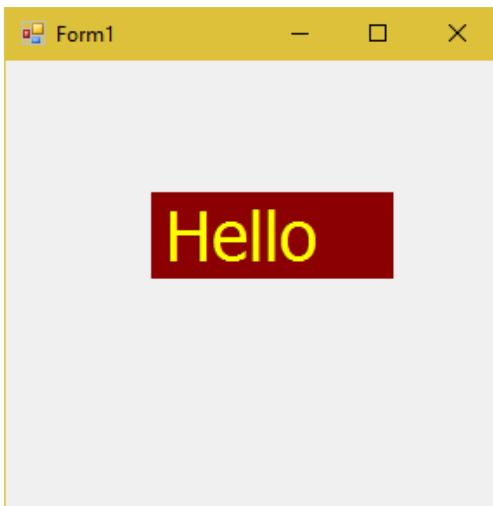
private void Lbl_MouseEnter(object sender, EventArgs e)
{
    lbl.BackColor = System.Drawing.Color.Yellow;
    lbl.ForeColor = System.Drawing.Color.DarkRed;
}

private void Lbl_MouseLeave(object sender, EventArgs e)
{
    lbl.BackColor = System.Drawing.Color.DarkRed;
    lbl.ForeColor = System.Drawing.Color.Yellow;
}
}
```

Running the Project

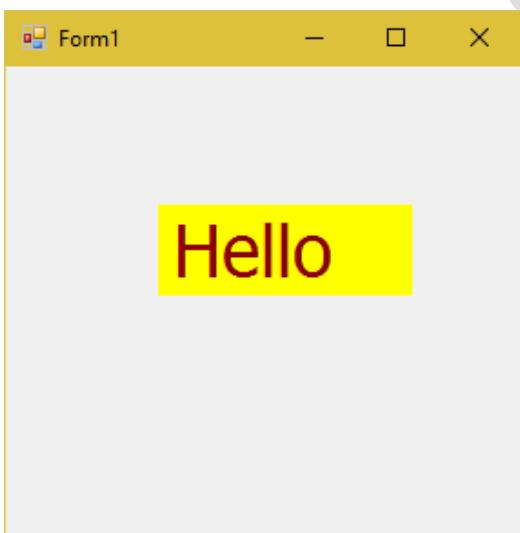
- Go to “Debug” menu and click on “Start Debugging”.

Output



When you place the mouse pointer on the label, background color and foreground colors will change.

When you click on the label it becomes as “Thanx”.



“System.Windows.Forms.Button” class

The “System.Windows.Forms.Button” class

- “Button” is a class; “System.Windows.Forms” is a namespace.
- The “Button” class / control is used to display an option on the windows form. Ex: OK, Cancel etc.

Steps for development of button:

- Import the namespace:
 - `using System.Windows.Forms;`

- Create a reference variable:
 - `Button referencevariable;`

- Create an object:
 - `referencevariable = new Button();`

- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event += method;`

- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "Button" class:

Sl. No	Property	Description
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <code><i>referencevariable.Text</i> = "text here";</code></p>

2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	AutoSize	<p><u>True:</u> The control will take the essential size automatically.</p> <p><u>False:</u> The control will not take the size automatically; we need to set the size manually.</p> <p><u>Syntax:</u> <code>referencevariable.AutoSize = true false;</code></p>
5	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
6	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
7	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
8	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>

9	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>
10	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
11	Image	<p>Displays an icon image inside the control.</p> <p><u>Options:</u> Any image (jpg, png etc).</p> <p><u>Syntax:</u> <code>referencevariable.Image = System.Drawing.Image.FromFile(string ImageFilePath);</code></p>
12	ImageAlign	<p>Represents the position of the image on the control.</p> <p><u>Options:</u> TopLeft TopCenter TopRight MiddleLeft MiddleCenter MiddleRight BottomLeft BottomCenter BottomRight.</p> <p><u>Syntax:</u> <code>referencevariable.ImageAlign = System.Drawing.ContentAlignment.TopLeft;</code></p>
13	.TextAlign	<p>Represents the position of the text on the control.</p> <p><u>Options:</u> TopLeft TopCenter TopRight MiddleLeft MiddleCenter MiddleRight BottomLeft BottomCenter BottomRight.</p> <p><u>Syntax:</u> <code>referencevariable.TextAlign = System.Drawing.ContentAlignment.TopLeft;</code></p>

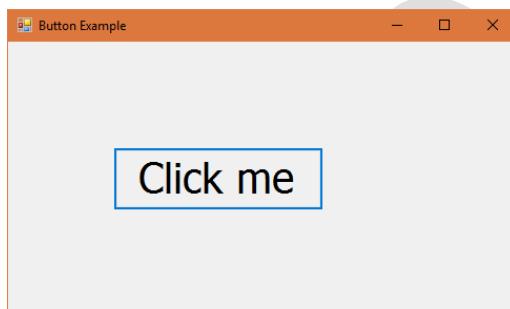
Events of "Button" class:

Sl. No	Event	Description

1	Click	Executes when the user clicks on the control. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>
2	MouseEnter	Executes when the user moves the mouse pointer from outside to inside the control. <u>Syntax:</u> <code>referencevariable.MouseEnter += methodname;</code>
3	MouseLeave	Executes when the user moves the mouse pointer from inside to outside the control. <u>Syntax:</u> <code>referencevariable.MouseLeave += methodname;</code>

“System.Windows.Forms.Button” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ButtonExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ButtonExample”.
- Click on OK.

- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace ButtonExample
{
    public partial class Form1 : Form
    {
        //create a reference variable
        Button button1;

        public Form1()
        {
            InitializeComponent();

            //form properties
            this.Size = new System.Drawing.Size(500, 300);
            this.Text = "Button Example";

            //create an object
            button1 = new Button();

            //set properties to the button
            button1.Text = "Click me please";
            button1.Font = new System.Drawing.Font("Tahoma", 30);
            button1.Size = new System.Drawing.Size(200, 60);
            button1.Location = new System.Drawing.Point(100, 100);
            button1.Cursor = Cursors.Hand;

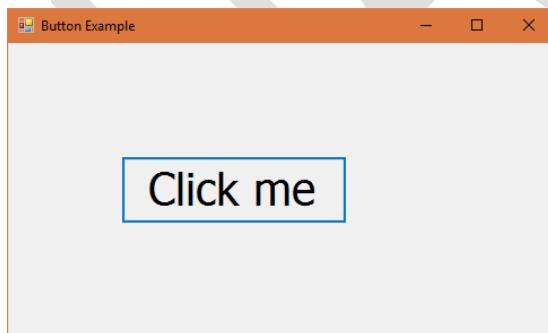
            //add events to the button
            button1.Click += button1_Click;
            button1.MouseEnter += Button1_MouseEnter;
```

```
button1.MouseLeave += Button1_MouseLeave;
//add button to the form
this.Controls.Add(button1);
}
private void button1_Click(object sender, EventArgs e)
{
    button1.Text = "Clicked";
}
private void Button1_MouseEnter(object sender, EventArgs e)
{
    button1.BackColor = System.Drawing.Color.LightGreen;
}
private void Button1_MouseLeave(object sender, EventArgs e)
{
    button1.BackColor = System.Drawing.SystemColors.Control;
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



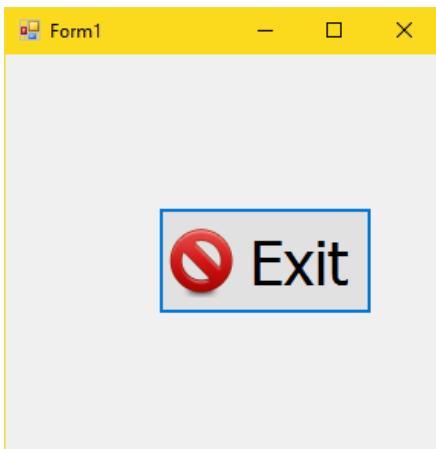
When you place the mouse pointer on the button, background color will change.

When you click on the button it becomes as “Thanx”.



"System.Windows.Forms.Button" class – with Image - Example

Expected Output



Creating Project

- Copy and paste “exit.png” file into “C:\CSharp” folder.
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ButtonWithImageExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ButtonWithImageExample”.

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace ButtonWithImageExample
{
    public partial class Form1 : Form
    {
        //create a reference variable
        Button button1;

        public Form1()
        {
            InitializeComponent();

            //create an object
            button1 = new Button();

            //set properties
            button1.Text = "Exit";
            button1.Font = new System.Drawing.Font("Tahoma", 30);
            button1.Size = new System.Drawing.Size(140, 70);
            button1.Location = new System.Drawing.Point(100, 100);
            button1.Cursor = Cursors.Hand;
            button1.Image =
                System.Drawing.Image.FromFile(@"C:\csharp\exit.png");
            button1.ImageAlign =
                System.Drawing.ContentAlignment.MiddleLeft;
            button1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

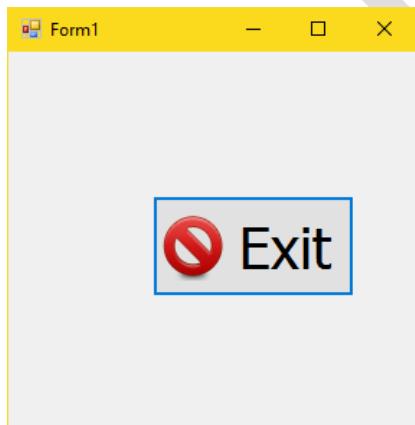
            //add "Click" event to the button
            button1.Click += button1_Click;
        }
    }
}
```

```
//add button to the form  
this.Controls.Add(button1);  
}  
  
private void button1_Click(object sender, EventArgs e)  
{  
    this.Close();  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

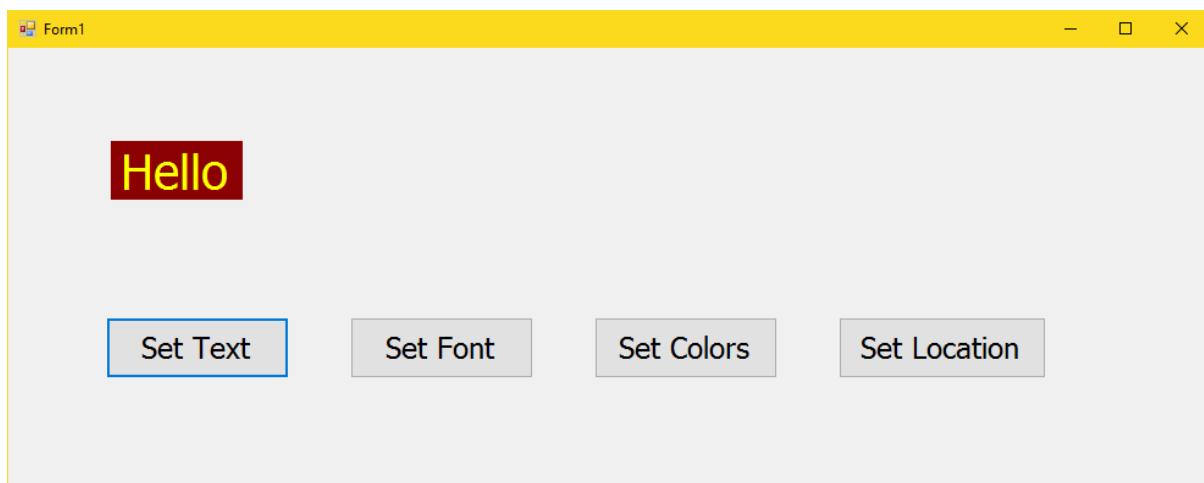
Output



Button appears with image.

When you click on “Exit” button, the form will be closed.

“System.Windows.Forms.Button” class – with Label - Example

Expected Output**Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ButtonWithLabelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ButtonWithLabelExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace ButtonWithLabelExample
{
    public partial class Form1 : Form
    {

```

```
//create reference variables
Label label1;
Button button1, button2, button3, button4;

public Form1()
{
    InitializeComponent();

    /*form properties*/
    this.Size = new Size(1000, 400);

    /* creating label1 */
    label1 = new Label();
    label1.Text = "Hello";
    label1.AutoSize = true;
    label1.Font = new Font("Tahoma", 30);
    label1.BackColor = Color.DarkRed;
    label1.ForeColor = Color.Yellow;
    label1.Location = new Point(84, 76);
    this.Controls.Add(label1);

    /* creating button1 */
    button1 = new Button();
    button1.Text = "Set Text";
    button1.Font = new Font("Tahoma", 18);
    button1.Size = new Size(150, 50);
    button1.Location = new Point(80, 220);
    button1.Click += button1_Click;
    this.Controls.Add(button1);

    /* creating button2 */
    button2 = new Button();
    button2.Text = "Set Font";
    button2.Font = new Font("Tahoma", 18);
    button2.Size = new Size(150, 50);
    button2.Location = new Point(280, 220);
    button2.Click += button2_Click;
```

```
this.Controls.Add(button2);

/* creating button3 */
button3 = new Button();
button3.Text = "Set Colors";
button3.Font = new Font("Tahoma", 18);
button3.Size = new Size(150, 50);
button3.Location = new Point(480, 220);
button3.Click += button3_Click;
this.Controls.Add(button3);

/* creating button4 */
button4 = new Button();
button4.Text = "Set Location";
button4.Font = new Font("Tahoma", 18);
button4.Size = new Size(170, 50);
button4.Location = new Point(680, 220);
button4.Click += button4_Click;
this.Controls.Add(button4);

}

private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "How are you";
}

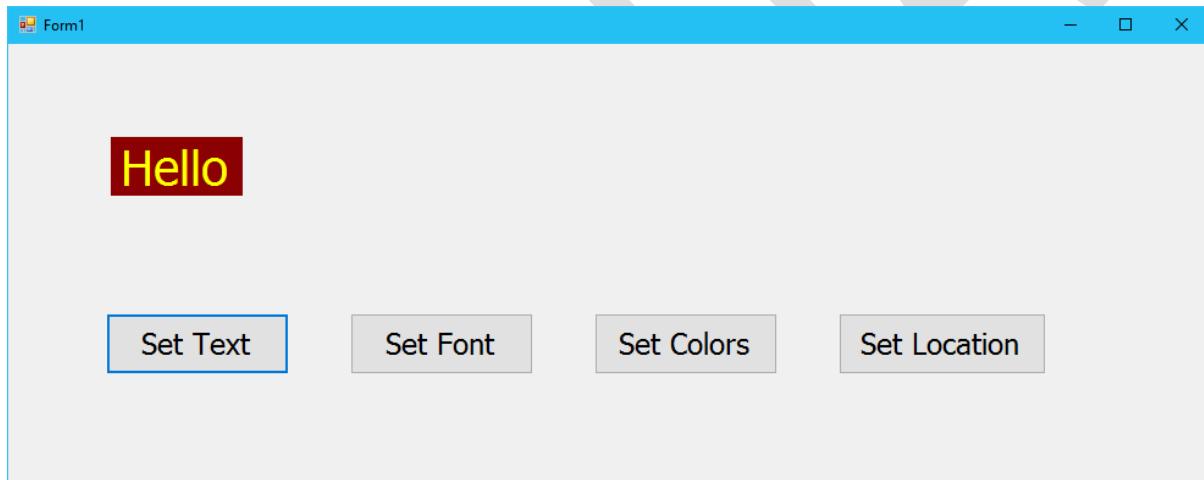
private void button2_Click(object sender, EventArgs e)
{
    label1.Font = new Font("Maiandra GD", 50, FontStyle.Bold);
}

private void button3_Click(object sender, EventArgs e)
{
    label1.BackColor = Color.LightGreen;
    label1.ForeColor = Color.Blue;
}
```

```
private void button4_Click(object sender, EventArgs e)
{
    label1.Location = new Point(300, 160);
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Click on the buttons and see the output.

“System.Windows.Forms.TextBox” class**The “System.Windows.Forms.TextBox” class**

- “TextBox” is a class; “System.Windows.Forms” is a namespace.
- The “TextBox” class / control is used to accept a string from the user. Ex: Username, password etc.

Steps for development of textbox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `TextBox referencevariable,`
- Create an object:
 - `referencevariable = new TextBox();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "TextBox" class:

Sl. No	Property	Description
1	Text	Represents text of the control. <u>Syntax:</u> <code><i>referencevariable.Text</i> = "text here";</code>
2	Font	Represents font settings of the control. <u>Syntax:</u> <code><i>referencevariable.Font</i> = new System.Drawing.Font(string <i>FontName</i>, int <i>FontSize</i>);</code>

3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <i>referencevariable.Size = new System.Drawing.Size(int width, int height);</i>
4	MaxLength	Represents the maximum number of characters that can be entered in the control. <u>Syntax:</u> <i>referencevariable.MaxLength = number;</i>
5	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <i>referencevariable.Location = new System.Drawing.Point(int x, int y);</i>
6	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <i>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</i>
7	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.BackColor = System.Drawing.Color.Green;</i>
8	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.ForeColor = System.Drawing.Color.Green;</i>

9	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
10	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
11	TextAlign	Represents the alignment of the text in the control. <u>Options:</u> Left Center Right <u>Syntax:</u> <i>referencevariable.TextAlign = System.Windows.Forms.HorizontalAlignment.Left;</i>
12	Readonly	<u>True:</u> The user can't modify the value of textbox. <u>False:</u> The user can modify the value of textbox. <u>Syntax:</u> <i>referencevariable.Readonly = true false;</i>
13	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>
14	AutoCompleteSource	Represents source of auto complete options. <u>Options:</u> FileSystem FileSystemDirectories CustomSource <u>Syntax:</u> <i>referencevariable.AutoCompleteSource = System.Windows.Forms.AutoCompleteSource.FileSystem;</i>

13	AutoCompleteMode	Represents how the auto complete options appear in the textbox, while typing. <u>Options:</u> None Suggest Append SuggestAppend <u>Syntax:</u> <i>referencevariable.AutoCompleteMode</i> = System.Windows.Forms.AutoCompleteMode.Suggest;
14	AutoCompleteCustomSource	Represents the string array, based on which the auto complete list should be appear while typing in the textbox. <u>Options:</u> any string array. <u>Syntax:</u> <i>referencevariable.AutoCompleteCustomSource</i> = new System.Windows.Forms.AutoCompleteStringCollection() { "value1", "value2", };

Events of "TextBox" class:

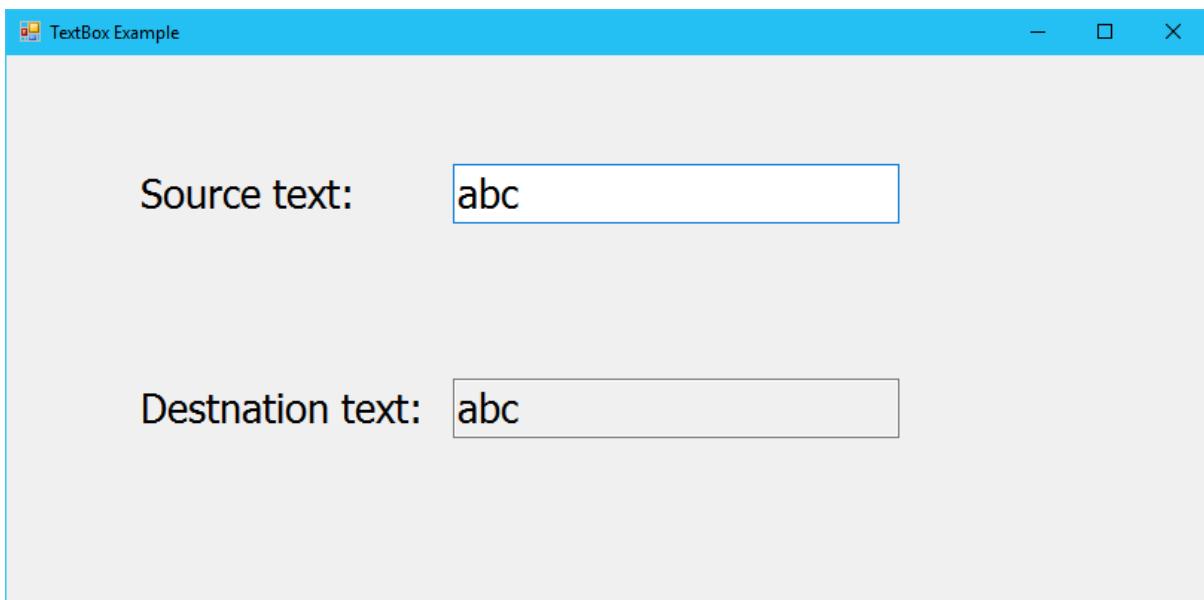
Sl. No	Event	Description
1	TextChanged	Executes when the user types a character in the control. <u>Syntax:</u> <i>referencevariable.TextChanged</i> += methodname;
2	Enter	Executes when the cursor enters into the control. <u>Syntax:</u> <i>referencevariable.Enter</i> += methodname;
3	Leave	Executes when the cursor leaves into the control. <u>Syntax:</u> <i>referencevariable.Leave</i> += methodname;
4	Click	Executes when the user clicks on the control. <u>Syntax:</u> <i>referencevariable.Click</i> += methodname;

5	KeyPress	Executes when the user presses any key on the keyboard (before accepting the character into the textbox). <u>Syntax:</u> <code>referencevariable.KeyPress += methodname;</code>
---	----------	--

Methods of "TextBox" class:

Sl. No	Method	Description
1	Clear()	Clears all text in the textbox. <u>Syntax:</u> <code>referencevariable.Clear();</code>
2	SelectAll()	Selects all text in the textbox. <u>Syntax:</u> <code>referencevariable.SelectAll();</code>
3	Cut()	Cuts the selected text in the textbox. <u>Syntax:</u> <code>referencevariable.Cut();</code>
4	Copy()	Copies the selected text in the textbox. <u>Syntax:</u> <code>referencevariable.Copy();</code>
5	Paste()	Pastes the cut / copied text in the textbox. <u>Syntax:</u> <code>referencevariable.Paste()</code>
6	Undo()	Undos the previous action in the textbox. <u>Syntax:</u> <code>referencevariable.Undo()</code>

"System.Windows.Forms.TextBox" class - Example**Expected Output**



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TextBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TextBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace TextBoxExample
{
    public partial class Form1 : Form
```

```
{  
    //create reference variables  
    Label label1, label2;  
    TextBox textbox1, textbox2;  
  
    public Form1()  
    {  
        InitializeComponent();  
  
        /* form properties */  
        this.Text = "TextBox Example";  
        this.Size = new Size(500, 250);  
        this.Font = new Font("Tahoma", 20);  
  
        /* creating label1 */  
        label1 = new Label();  
        label1.Text = "Source text:";  
        label1.AutoSize = true;  
        label1.Location = new Point(84, 76);  
        this.Controls.Add(label1);  
  
        /* creating label2 */  
        label2 = new Label();  
        label2.Text = "Destination text:";  
        label2.AutoSize = true;  
        label2.Location = new Point(84, 220);  
        this.Controls.Add(label2);  
  
        /* creating textbox1 */  
        textbox1 = new TextBox();  
        textbox1.Size = new Size(300, 50);  
        textbox1.Location = new Point(300, 73);  
        textbox1.MaxLength = 30;  
        textbox1.TextChanged += Textbox1_TextChanged;  
        this.Controls.Add(textbox1);  
  
        /* creating textbox2 */
```

```
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(300, 217);
textbox2.ReadOnly = true;
this.Controls.Add(textbox2);

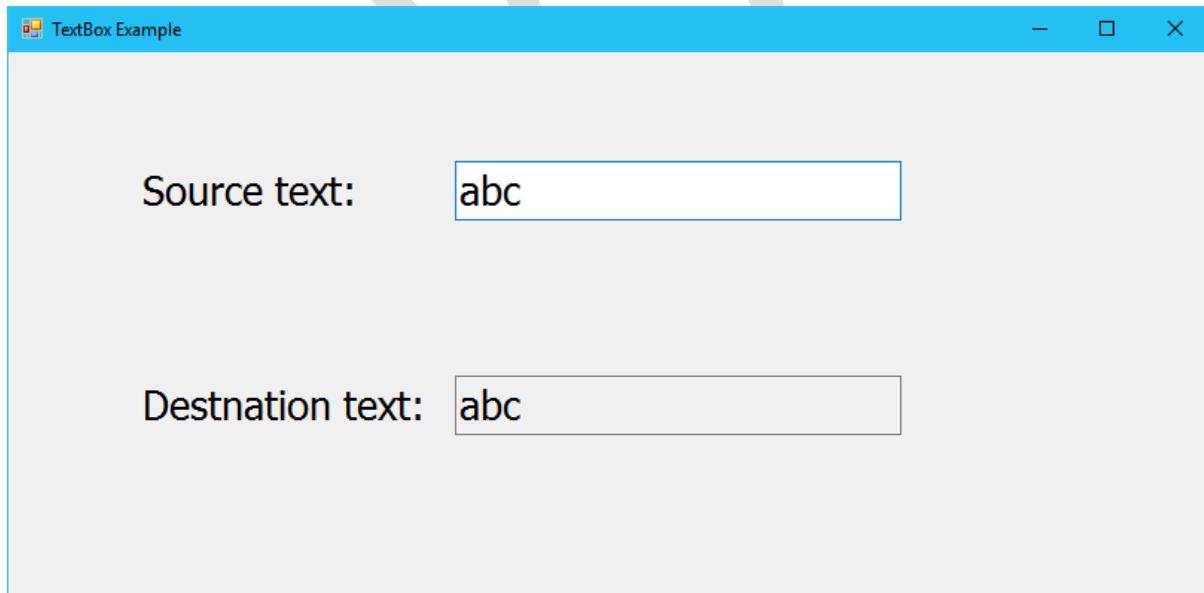
}

private void Textbox1_TextChanged(object sender, EventArgs e)
{
    textbox2.Text = textbox1.Text;
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

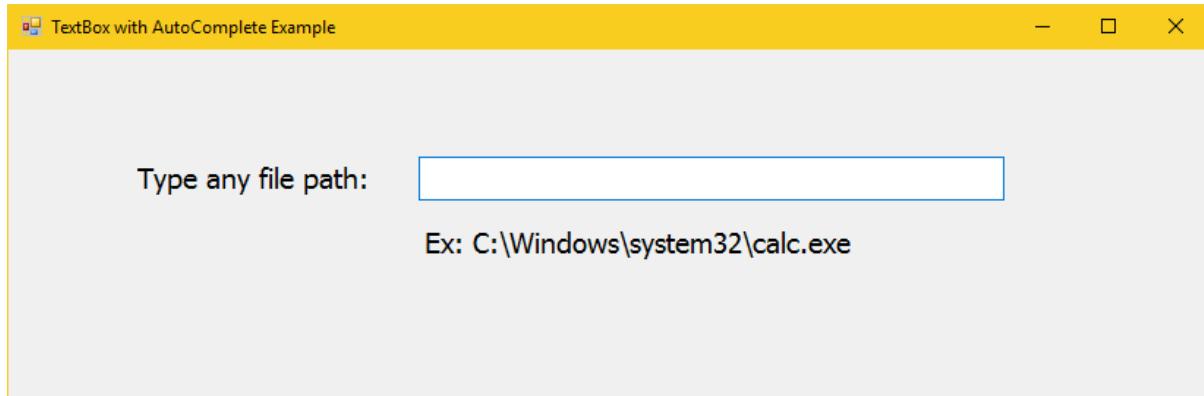
Output



When you type something in the first textbox, it automatically copied into the second textbox.

“System.Windows.Forms.TextBox” class – With AutoComplete - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “AutoCompleteExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AutoCompleteExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

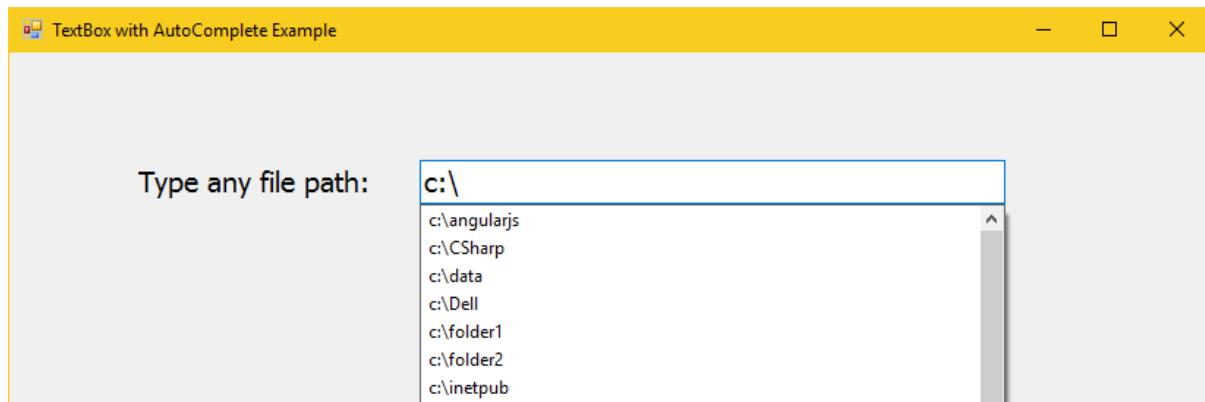
namespace AutoCompleteExample
{
    public partial class Form1 : Form
```

```
{  
    Label label1, label2;  
    TextBox textbox1;  
  
    public Form1()  
    {  
        InitializeComponent();  
  
        this.Text = "TextBox with AutoComplete Example";  
        this.Size = new Size(1000, 300);  
        this.Font = new Font("Tahoma", 14);  
  
        /* creating label1 */  
        label1 = new Label();  
        label1.Text = "Type any file path: ";  
        label1.AutoSize = true;  
        label1.Location = new Point(84, 76);  
        this.Controls.Add(label1);  
  
        /* creating textbox1 */  
        textbox1 = new TextBox();  
        textbox1.Size = new Size(400, 50);  
        textbox1.Location = new Point(280, 73);  
        textbox1.AutoCompleteSource = AutoCompleteSource.FileSystem;  
        textbox1.AutoCompleteMode = AutoCompleteMode.Suggest;  
        this.Controls.Add(textbox1);  
  
        /* creating label2 */  
        label2 = new Label();  
        label2.Text = @"Ex: C:\Windows\system32\calc.exe";  
        label2.AutoSize = true;  
        label2.Location = new Point(280, 120);  
        this.Controls.Add(label2);  
    }  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

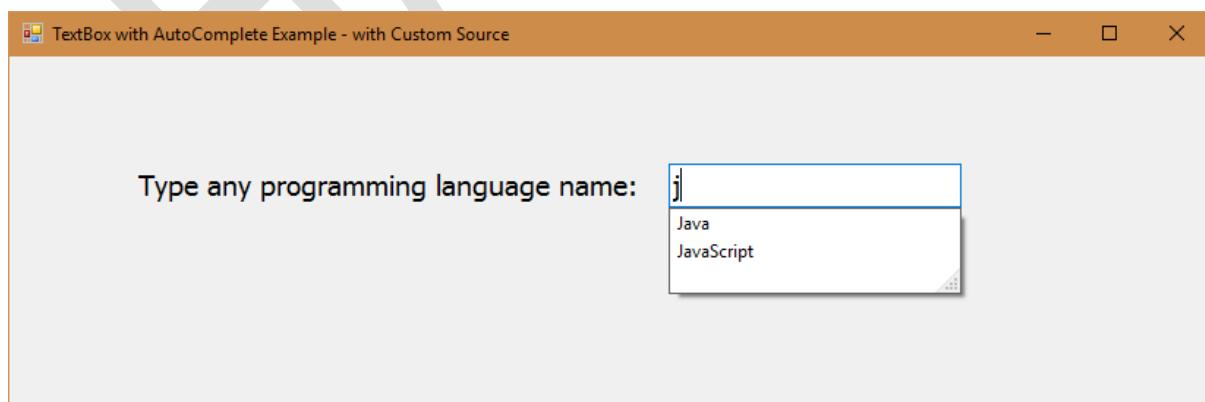
Output



When you start typing a file path in the textbox, it automatically shows all folders and files.

“System.Windows.Forms.TextBox” class – With AutoComplete – Example 2

Expected Output



Creating Project

- Open Visual Studio 2019.

- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “AutoCompleteExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AutoCompleteExample2”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace AutoCompleteExample2
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        TextBox textbox1;

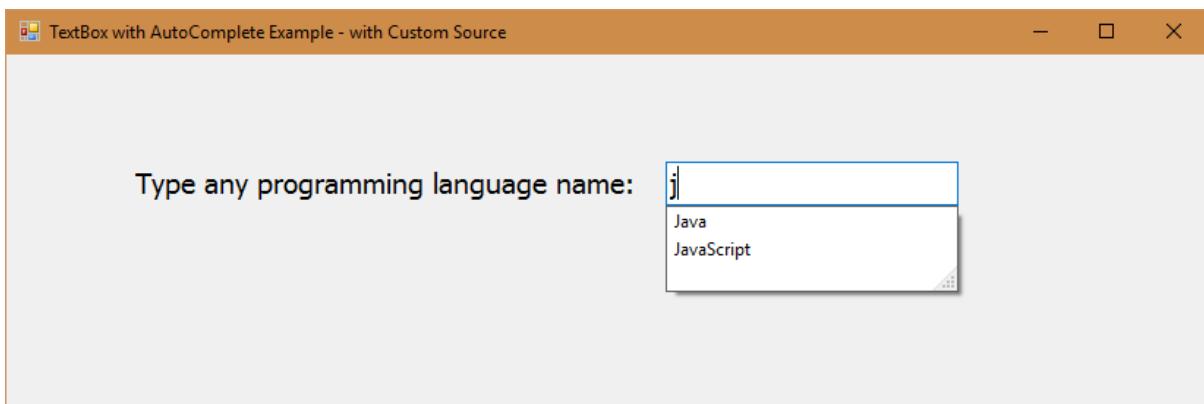
        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Text = "TextBox with AutoComplete Example - with Custom
Source";
            this.Size = new Size(1000, 300);
            this.Font = new Font("Tahoma", 14);
            /* creating label1 */
            label1 = new Label();
```

```
label1.Text = "Type any programming language name:";  
label1.AutoSize = true;  
label1.Location = new Point(84, 76);  
this.Controls.Add(label1);  
/* creating textBox1 */  
textBox1 = new TextBox();  
textBox1.Size = new Size(200, 50);  
textBox1.Location = new Point(450, 73);  
textBox1.AutoCompleteSource =  
AutoCompleteSource.CustomSource;  
textBox1.AutoCompleteCustomSource = new  
AutoCompleteStringCollection() { "ActionScript", "AppleScript", "Asp",  
"BASIC", "C", "C++", "Clojure", "COBOL", "ColdFusion", "Erlang", "Fortran",  
"Groovy", "Haskell", "Java", "JavaScript", "Lisp", "Perl", "PHP", "Python",  
"Ruby", "Scala", "Scheme" };  
textBox1.AutoCompleteMode = AutoCompleteMode.Suggest;  
this.Controls.Add(textBox1);  
/* creating label2 */  
label2 = new Label();  
label2.Text = @"Ex: Java";  
label2.AutoSize = true;  
label2.Location = new Point(450, 120);  
this.Controls.Add(label2);  
}  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

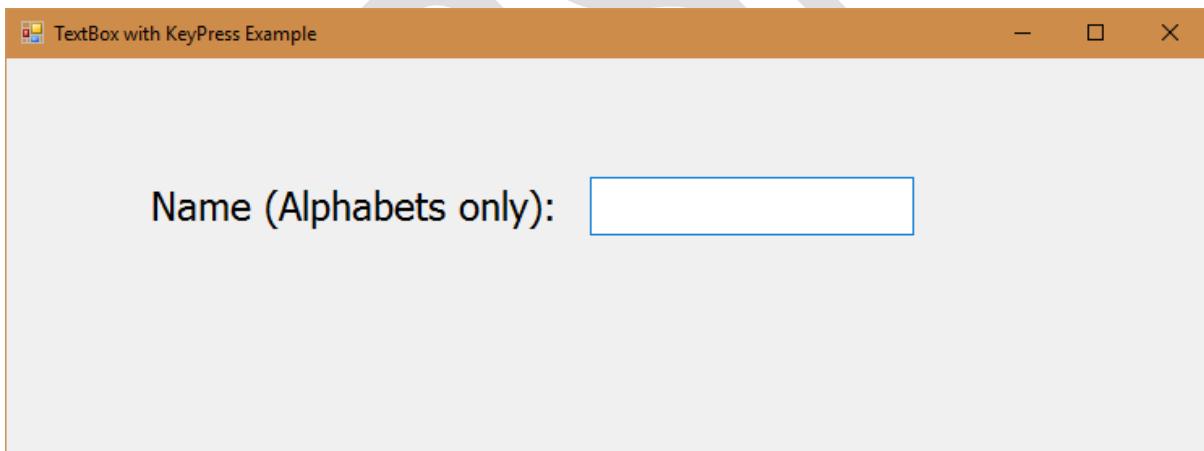
Output



When you start typing a character in the textbox, it automatically shows the matching names.

"System.Windows.Forms.TextBox" class – With KeyPress – Example

Expected Output



Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “KeyPressEventExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “KeyPressEventExample”.

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace KeyPressEventExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1;
        TextBox textbox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "TextBox with KeyPress Example";
            this.Size = new Size(700, 250);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Name (Alphabets only): ";
            label1.AutoSize = true;
            label1.Location = new Point(84, 76);
            this.Controls.Add(label1);

            /* creating textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(200, 50);
```

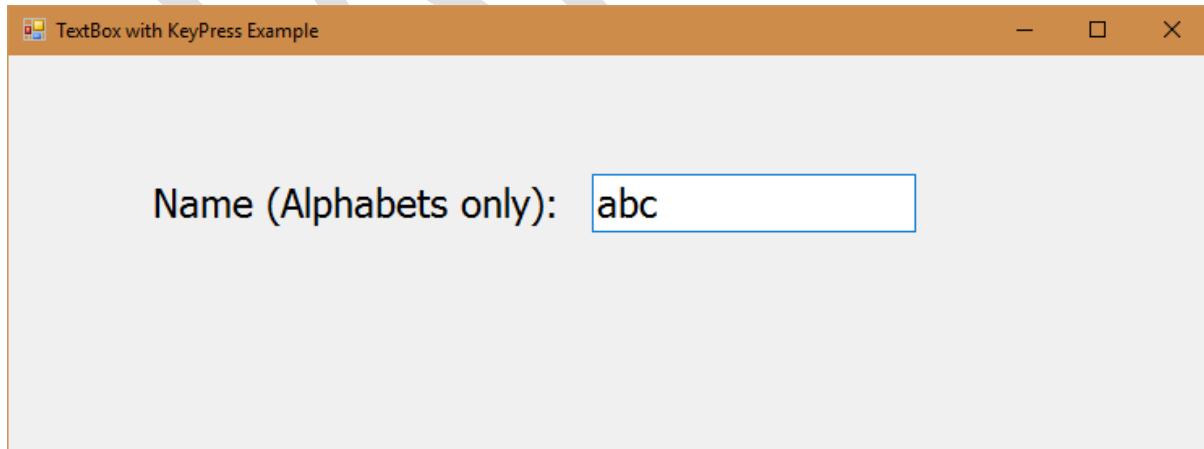
```
textbox1.Location = new Point(360, 73);
textBox1.KeyPress += textBox1_KeyPress;
this.Controls.Add(textBox1);
}

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    char ch = e.KeyChar;
    if ((ch >= 65 && ch <= 90) || (ch >= 97 && ch <= 122) || (ch == 32) || (ch ==
127) || (ch == 8) || (ch == 9))
        e.Handled = false; //accept the currently pressed character
    else
        e.Handled = true; //reject the currently pressed character
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

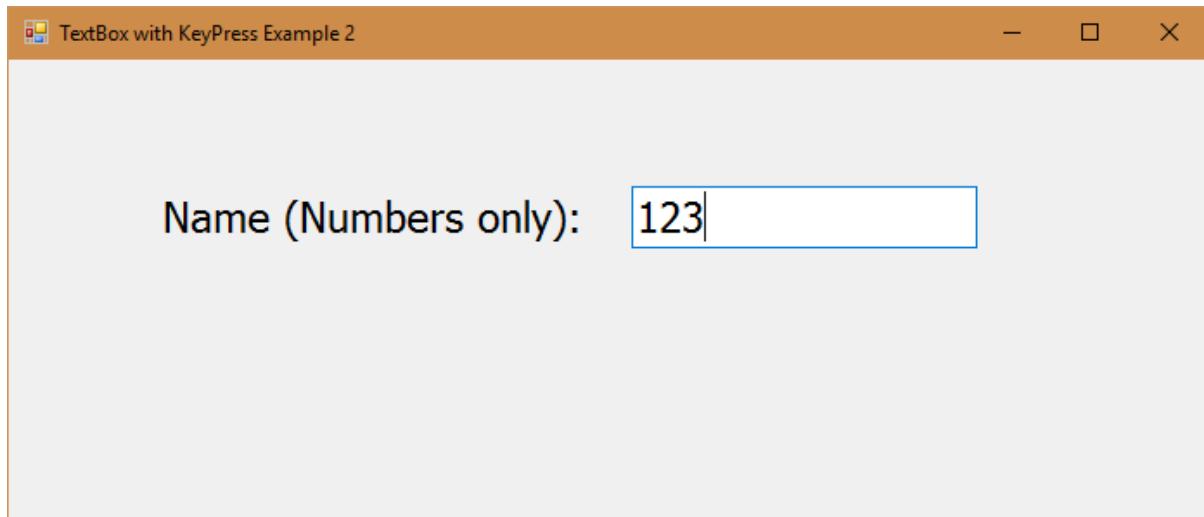


You can type only alphabets in the textbox.

If you type numbers and special symbols, it will not be accepted in the textbox.

“System.Windows.Forms.TextBox” class – With KeyPress – Example 2

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “KeyPressEventExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “KeyPressEventExample2”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
```

```
namespace KeyPressEventExample2
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1;
        TextBox textbox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "TextBox with KeyPress Example 2";
            this.Size = new Size(550, 250);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Name (Numbers only): ";
            label1.AutoSize = true;
            label1.Location = new Point(84, 76);
            this.Controls.Add(label1);

            /* creating textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(200, 50);
            textbox1.Location = new Point(360, 73);
            textbox1.KeyPress += textBox1_KeyPress;
            this.Controls.Add(textbox1);
        }

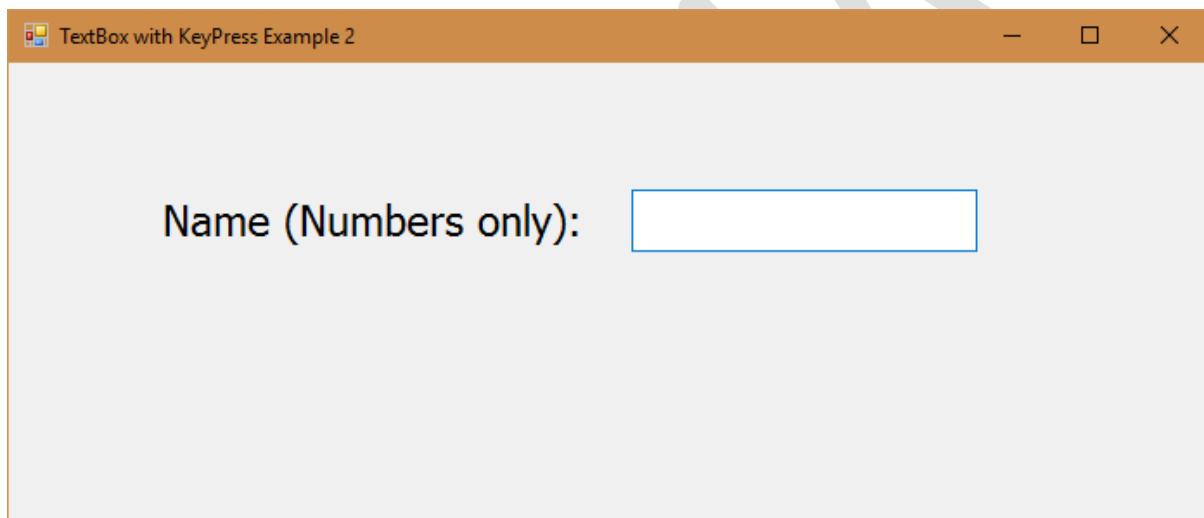
        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
            char ch = e.KeyChar;
            if ((ch >= 48 && ch <= 57) || (ch == 127) || (ch == 8) || (ch == 9))
                e.Handled = false;
        }
    }
}
```

```
    else
        e.Handled = true;
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

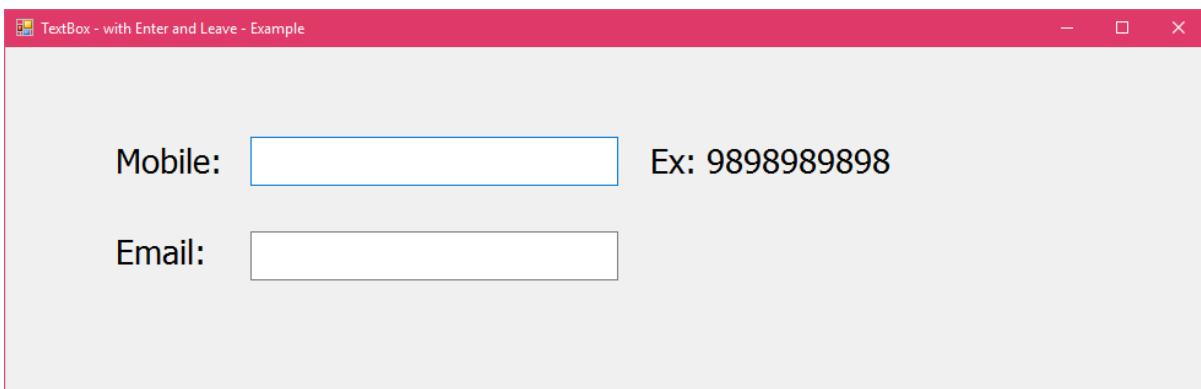


You can type only numbers in the textbox.

If you type alphabets and special symbols, it will not be accepted in the textbox.

“System.Windows.Forms.TextBox” class – With Enter and Leave – Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “EnterLeaveExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EnterLeaveExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace EnterLeaveExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2, label3, label4;
```

```
TextBox textbox1, textbox2;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Text = "TextBox - with Enter and Leave - Example";
    this.Size = new Size(800, 250);
    this.Font = new Font("Tahoma", 20);

    /* creating label1 */
    label1 = new Label();
    label1.Text = "Mobile:";
    label1.AutoSize = true;
    label1.Location = new Point(84, 76);
    this.Controls.Add(label1);

    /* creating label2 */
    label2 = new Label();
    label2.Text = "Email:";
    label2.AutoSize = true;
    label2.Location = new Point(84, 150);
    this.Controls.Add(label2);

    /* creating textbox1 */
    textbox1 = new TextBox();
    textbox1.Size = new Size(300, 50);
    textbox1.Location = new Point(200, 73);
    textbox1.Enter += Textbox1_Enter;
    textbox1.Leave += Textbox1_Leave;
    this.Controls.Add(textbox1);

    /* creating textbox2 */
    textbox2 = new TextBox();
    textbox2.Size = new Size(300, 50);
    textbox2.Location = new Point(200, 150);
    textbox2.Enter += Textbox2_Enter;
    textbox2.Leave += Textbox2_Leave;
    this.Controls.Add(textbox2);
```

```
/* creating label3 */
label3 = new Label();
label3.Text = "Ex: 9898989898";
label3.AutoSize = true;
label3.Location = new Point(520, 76);
label3.Visible = false;
this.Controls.Add(label3);

/* creating label4 */
label4 = new Label();
label4.Text = "Ex: sample@gmail.com";
label4.AutoSize = true;
label4.Location = new Point(520, 150);
label4.Visible = false;
this.Controls.Add(label4);

}

private void Textbox1_Enter(object sender, EventArgs e)
{
    label3.Visible = true;
}

private void Textbox1_Leave(object sender, EventArgs e)
{
    label3.Visible = false;
}

private void Textbox2_Enter(object sender, EventArgs e)
{
    label4.Visible = true;
}

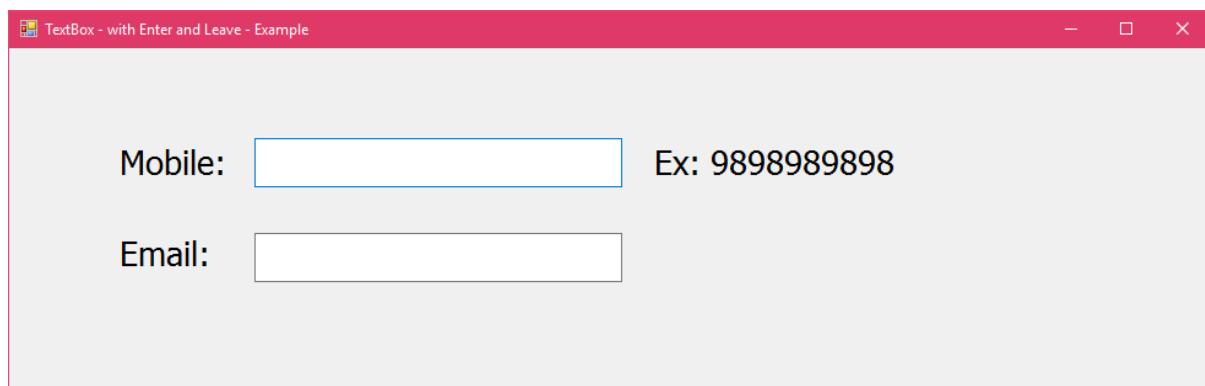
private void Textbox2_Leave(object sender, EventArgs e)
{
    label4.Visible = false;
}
```

Running the Project

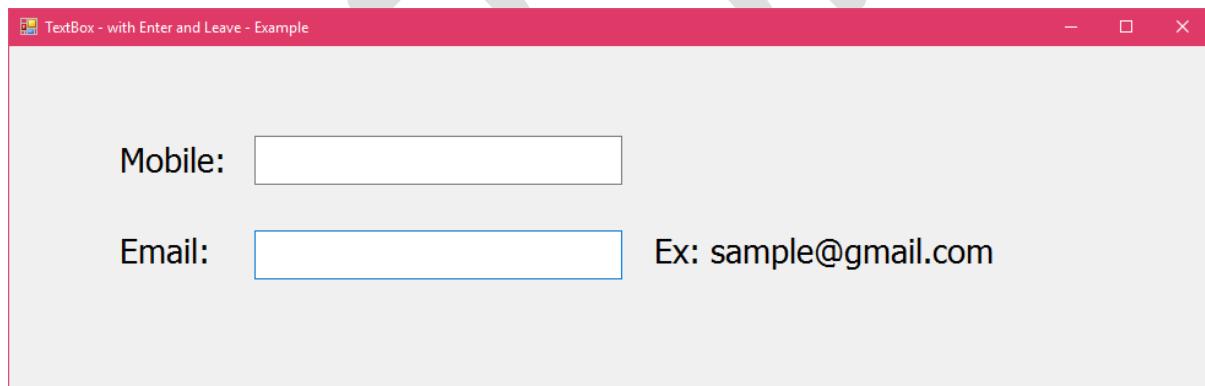
- Go to “Debug” menu and click on “Start Debugging”.

Output

When the cursor is inside mobile textbox, it shows example for mobile number.

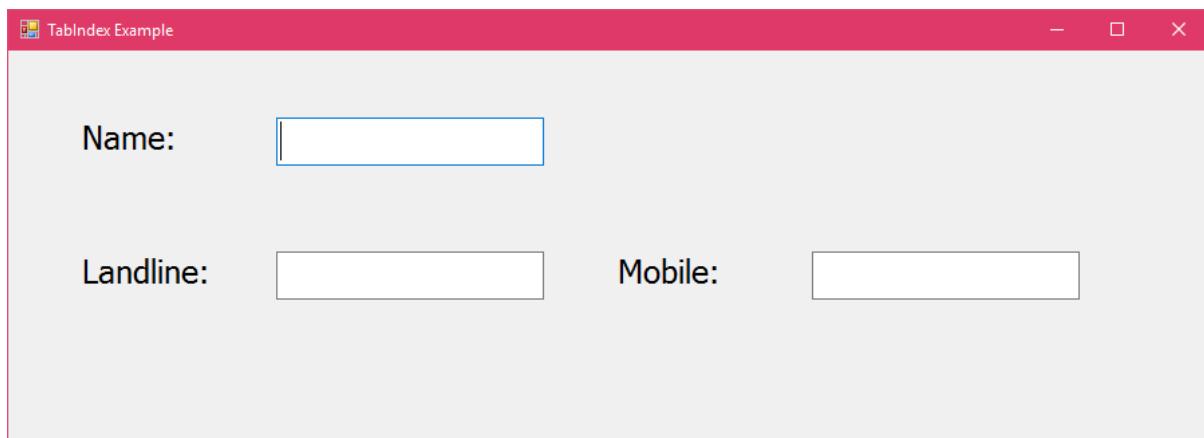


When the cursor is inside email textbox, it shows example for email.



“System.Windows.Forms.TextBox” class – With TabIndex – Example

Expected Output

**Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TabIndexExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TabIndexExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```

using System;
using System.Windows.Forms;
using System.Drawing;

namespace TabIndexExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2, label3;
    }
}

```

```
TextBox textbox1, textbox2, textbox3;
```

```
public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Text = "TabIndex Example";
    this.Size = new Size(430, 170);
    this.Font = new Font("Tahoma", 18);

    /* creating label1 */
    label1 = new Label();
    label1.Text = "Name: ";
    label1.AutoSize = true;
    label1.Location = new Point(50, 50);
    this.Controls.Add(label1);

    /* creating label2 */
    label2 = new Label();
    label2.Text = "Landline: ";
    label2.AutoSize = true;
    label2.Location = new Point(50, 150);
    this.Controls.Add(label2);

    /* creating label3 */
    label3 = new Label();
    label3.Text = "Mobile: ";
    label3.AutoSize = true;
    label3.Location = new Point(450, 150);
    this.Controls.Add(label3);

    /* creating textbox1 */
    textbox1 = new TextBox();
    textbox1.Size = new Size(200, 30);
    textbox1.Location = new Point(200, 50);
    textbox1.TabIndex = 1;
```

```
this.Controls.Add(textBox1);

/* creating textBox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(200, 30);
textBox2.Location = new Point(200, 150);
textBox2.TabIndex = 3;
this.Controls.Add(textBox2);

/* creating textBox3 */
textBox3 = new TextBox();
textBox3.Size = new Size(200, 30);
textBox3.Location = new Point(600, 150);
textBox3.TabIndex = 2;
this.Controls.Add(textBox3);

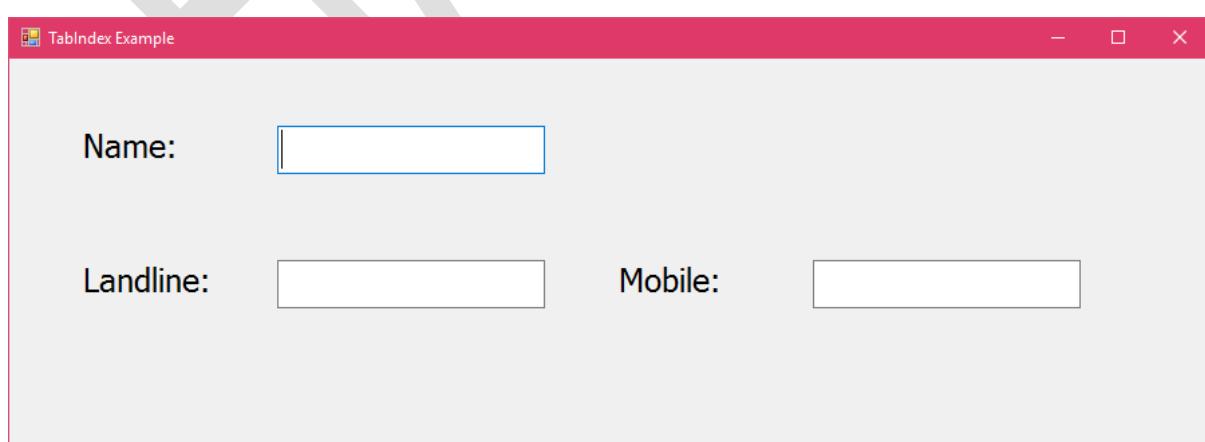
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

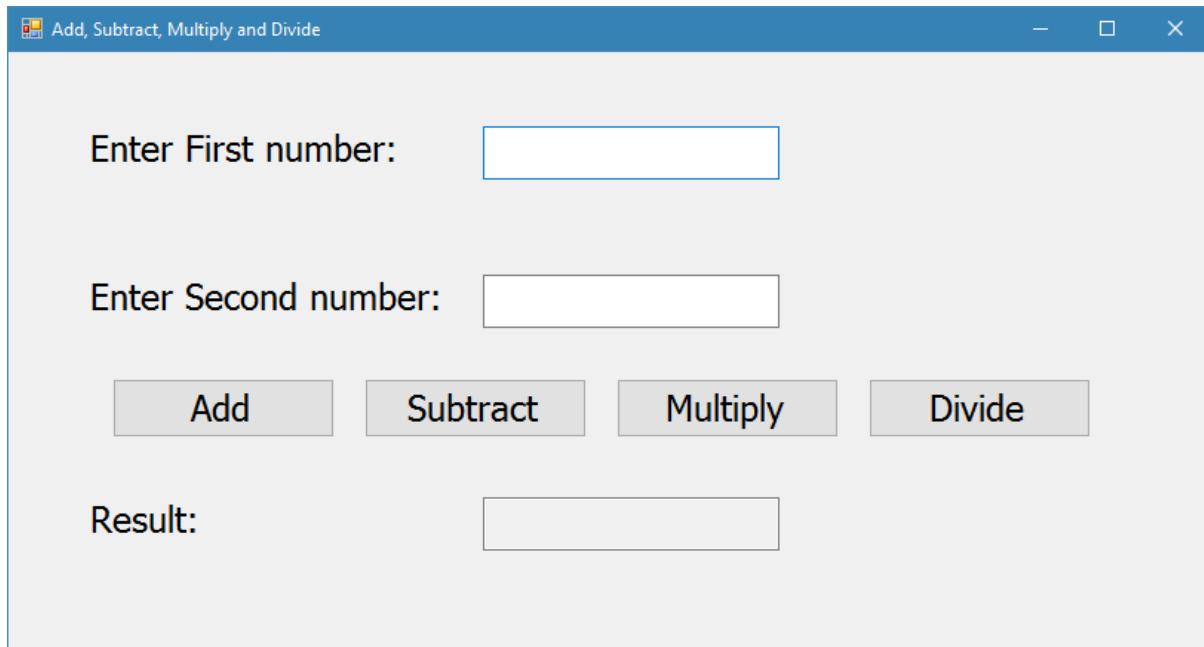
Output

The cursor sequence will be: Name, Mobile and then Landline.



“System.Windows.Forms.TextBox” class – Math – Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “AddSubtractMultiplyDivideExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “AddSubtractMultiplyDivideExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
```

```
using System.Windows.Forms;
using System.Drawing;

namespace AddSubtractMultiplyDivideExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1, button2, button3, button4;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "Add, Subtract, Multiply and Divide";
            this.Size = new Size(390, 220);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Enter First number: ";
            label1.AutoSize = true;
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* creating label2 */
            label2 = new Label();
            label2.Text = "Enter Second number: ";
            label2.AutoSize = true;
            label2.Location = new Point(50, 150);
            this.Controls.Add(label2);

            /* creating label3 */
            label3 = new Label();
```

```
label3.Text = "Result: ";
label3.AutoSize = true;
label3.Location = new Point(50, 300);
this.Controls.Add(label3);

/* creating textBox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(200, 30);
textBox1.Location = new Point(320, 50);
this.Controls.Add(textBox1);

/* creating textBox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(200, 30);
textBox2.Location = new Point(320, 150);
this.Controls.Add(textBox2);

/* creating textBox3 */
textBox3 = new TextBox();
textBox3.Size = new Size(200, 30);
textBox3.Location = new Point(320, 300);
textBox3.ReadOnly = true;
this.Controls.Add(textBox3);

/* creating button1 */
button1 = new Button();
button1.Size = new Size(150, 40);
button1.Text = "Add";
button1.Location = new Point(70, 220);
button1.Click += Button1_Click;
this.Controls.Add(button1);

/* creating button2 */
button2 = new Button();
button2.Size = new Size(150, 40);
button2.Text = "Subtract";
```

```
button2.Location = new Point(240, 220);
button2.Click += Button2_Click;
this.Controls.Add(button2);

/* creating button3 */
button3 = new Button();
button3.Size = new Size(150, 40);
button3.Text = "Multiply";
button3.Location = new Point(410, 220);
button3.Click += Button3_Click;
this.Controls.Add(button3);

/* creating button4 */
button4 = new Button();
button4.Size = new Size(150, 40);
button4.Text = "Divide";
button4.Location = new Point(580, 220);
button4.Click += Button4_Click;
this.Controls.Add(button4);

}

private void Button1_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int b = Convert.ToInt32(textBox2.Text);
    int c = a + b;
    textBox3.Text = Convert.ToString(c);
}

private void Button2_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int b = Convert.ToInt32(textBox2.Text);
    int c = a - b;
    textBox3.Text = Convert.ToString(c);
}
```

```

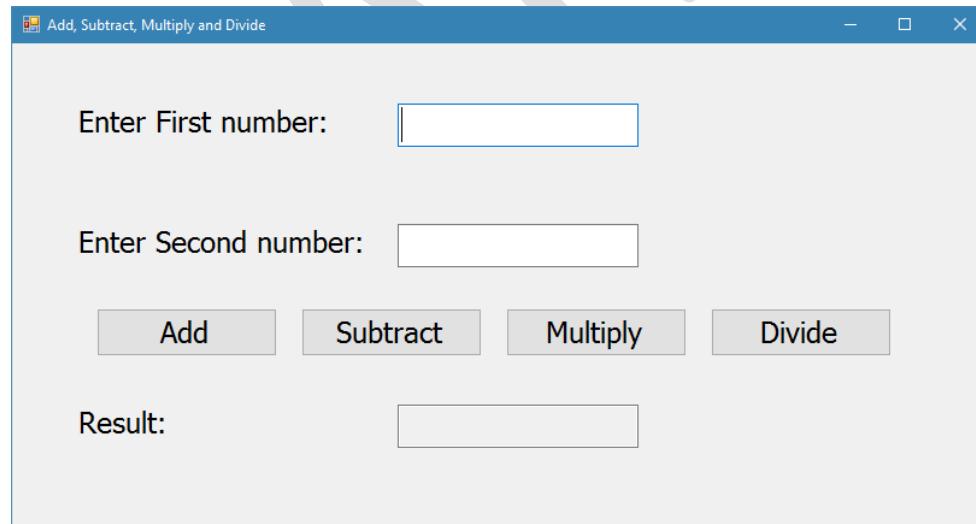
private void Button3_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int b = Convert.ToInt32(textBox2.Text);
    int c = a * b;
    textBox3.Text = Convert.ToString(c);
}

private void Button4_Click(object sender, EventArgs e)
{
    int a = Convert.ToInt32(textBox1.Text);
    int b = Convert.ToInt32(textBox2.Text);
    int c = a / b;
    textBox3.Text = Convert.ToString(c);
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Enter any first number and second number and click on “Add”, “Subtract”, “Multiply” and “Divide” buttons.

“System.Windows.Forms. NumericUpDown” class

The “System.Windows.Forms.NumericUpDown” class

- “NumericUpDown” is a class; “System.Windows.Forms” is a namespace.
- The “NumericUpDown” class / control is used to accept a number from the user. It doesn’t accept alphabets.

Steps for development of NumericUpDown:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `NumericUpDown referencevariable;`
- Create an object:
 - `referencevariable = new NumericUpDown();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event = method;`
- Add control to the form:

- `this.Controls.Add(referencevariable);`

Properties of "NumericUpDown" class:

Sl. No	Property	Description
1	Value	Represents the current value of the control. <u>Syntax:</u> <code>referencevariable.Value = any number;</code>
2	Minimum	Represents the minimum value the control. <u>Syntax:</u> <code>referencevariable.Minimum = any number;</code>
3	Maximum	Represents the maximum value the control. <u>Syntax:</u> <code>referencevariable.Maximum = any number;</code>
4	Increment	Represents the increment / decrement value the control. <u>Syntax:</u> <code>referencevariable.Increment = any number;</code>
5	DecimalPlaces	Represents the no. of decimal places. <u>Syntax:</u> <code>referencevariable.DecimalPlaces = any number;</code>
6	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
7	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>

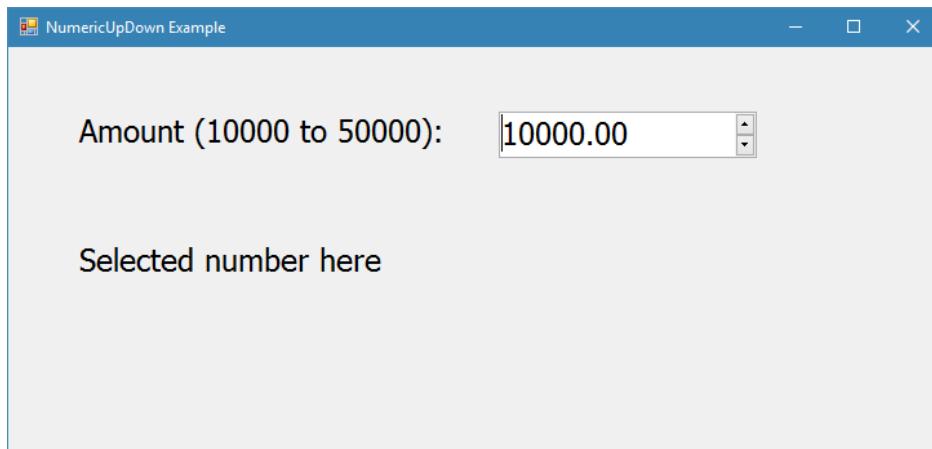
8	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
9	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
10	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
11	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
12	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>
13	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
14	TextAlign	<p>Represents the alignment of the text in the control.</p> <p><u>Options:</u> Left Center Right</p> <p><u>Syntax:</u> <code>referencevariable.TextAlign = System.Windows.Forms.HorizontalAlignment.Left;</code></p>

15	Readonly	<u>True:</u> The user can't modify the value of control. <u>False:</u> The user can modify the value of control. <u>Syntax:</u> <i>referencevariable.Readonly = true false;</i>
16	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>

Events of "NumericUpDown" class:

Sl. No	Event	Description
1	ValueChanged	Executes when the user types some number in the control. <u>Syntax:</u> <i>referencevariable.ValueChanged += methodname;</i>
2	Enter	Executes when the cursor enters into the control. <u>Syntax:</u> <i>referencevariable.Enter += methodname;</i>
3	Leave	Executes when the cursor leaves into the control. <u>Syntax:</u> <i>referencevariable.Leave += methodname;</i>
4	Click	Executes when the user clicks on the control. <u>Syntax:</u> <i>referencevariable.Click += methodname;</i>
5	KeyPress	Executes when the user presses any key on the keyboard (before accepting the character into the control). <u>Syntax:</u> <i>referencevariable.KeyPress += methodname;</i>

"System.Windows.Forms. NumericUpDown" class – Example**Expected Output**



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “NumericUpDownExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NumericUpDownExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace NumericUpDownExample
{
    public partial class Form1 : Form
    {
```

```
//create reference variables
Label label1, label2;
NumericUpDown numericupdown1;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Text = "NumericUpDown Example";
    this.Size = new Size(350, 180);
    this.Font = new Font("Tahoma", 18);

    /* creating label1 */
    label1 = new Label();
    label1.Text = "Amount (10000 to 50000):";
    label1.AutoSize = true;
    label1.Location = new Point(50, 50);
    this.Controls.Add(label1);

    /* creating label2 */
    label2 = new Label();
    label2.Text = "Selected number here";
    label2.AutoSize = true;
    label2.Location = new Point(50, 150);
    this.Controls.Add(label2);

    /* creating numericupdown1 */
    numericupdown1 = new NumericUpDown();
    numericupdown1.Size = new Size(200, 30);
    numericupdown1.Location = new Point(380, 50);
    numericupdown1.Minimum = 10000;
    numericupdown1.Maximum = 50000;
    numericupdown1.Increment = 100;
    numericupdown1.DecimalPlaces = 2;
    numericupdown1.ValueChanged += Numericupdown1_ValueChanged;
    this.Controls.Add(numericupdown1);
}
```

```

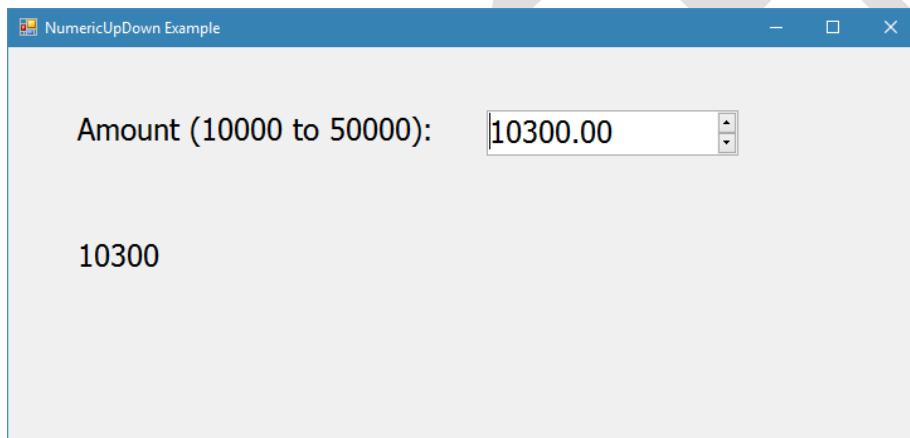
private void Numericupdown1_ValueChanged(object sender,
EventArgs e)
{
    decimal n = numericupdown1.Value;
    label2.Text = Convert.ToString(n);
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



"System.Windows.Forms. DateTimePicker" class

- “DateTimePicker” is a class; “System.Windows.Forms” is a namespace.
- The “DateTimePicker” class / control is used to accept a date or time from the user. It automatically shows popup calendar.

Steps for development of DateTimePicker:

- Import the namespace:

- `using System.Windows.Forms;`

- Create a reference variable:
 - `DateTimePicker referencevariable,`

- Create an object:
 - `referencevariable = new DateTimePicker();`

- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event = method;`

- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "DateTimePicker" class:

Sl. No	Property	Description
1	Value	<p>Represents the currently selected date & time of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Value = any number;</code></p>
2	Format	<p>Represents the date & time format of the control.</p> <p><u>Options:</u> Short Long Time Custom</p> <p><u>Syntax:</u> <code>referencevariable.Format = System.Windows.Forms.DateTimePickerFormat.any option;</code></p>

3	CustomFormat	Represents the custom format for the date / time. <u>Options:</u> “M/d/yyyy” “yyyy/M/d” “d/M/yyyy” etc. <u>Syntax:</u> <code>referencevariable.CustomFormat = any option;</code>
4	MinDate	Represents the minimum date that can be selected in the control. <u>Syntax:</u> <code>referencevariable.MinDate = any date;</code>
5	MaxDate	Represents the maximum date that can be selected in the control. <u>Syntax:</u> <code>referencevariable.MaxDate = any date;</code>
6	ShowUpDown	<u>True:</u> Shows up / down buttons, instead of popup calendar. <u>False:</u> Doesn't show the up / down buttons, show popup calendar normally. <u>Syntax:</u> <code>referencevariable.ShowUpDown = true false;</code>
7	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
8	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
9	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code>

10	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <i>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</i>
11	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.BackColor = System.Drawing.Color.Green;</i>
12	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.ForeColor = System.Drawing.Color.Green;</i>
13	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
14	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
15	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>

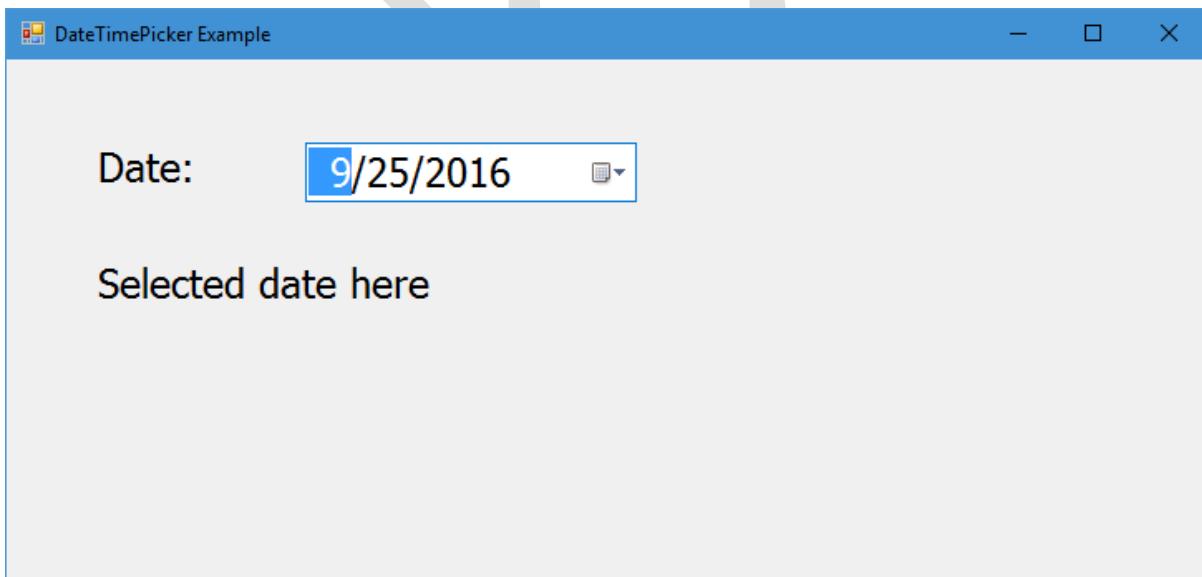
Events of "DateTimePicker" class:

Sl. No	Event	Description
1	ValueChanged	Executes when the user selects a date in the control. <u>Syntax:</u> <i>referencevariable.ValueChanged += methodname;</i>

2	Enter	Executes when the cursor enters into the control. <u>Syntax:</u> <code>referencevariable.Enter += methodname;</code>
3	Leave	Executes when the cursor leaves from the control. <u>Syntax:</u> <code>referencevariable.Leave += methodname;</code>
4	KeyPress	Executes when the user presses any key on the keyboard (before accepting the character into the control). <u>Syntax:</u> <code>referencevariable.KeyPress += methodname;</code>

"System.Windows.Forms. DateTimePicker" class – Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DateTimePickerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DateTimePickerExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace DateTimePickerExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        DateTimePicker datetimetypepicker1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "DateTimePicker Example";
            this.Size = new Size(350, 180);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Date: ";
        }
}
```

```

label1.AutoSize = true;
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* creating label2 */
label2 = new Label();
label2.Text = "Selected date here";
label2.AutoSize = true;
label2.Location = new Point(50, 120);
this.Controls.Add(label2);

/* creating datetimetype1 */
datetimetype1 = new DateTimePicker();
datetimetype1.Size = new Size(200, 30);
datetimetype1.Location = new Point(180, 50);
datetimetype1.Format = DateTimePickerFormat.Short;
datetimetype1.ValueChanged += Datetimetype1_ValueChanged;
this.Controls.Add(datetimetype1);

}

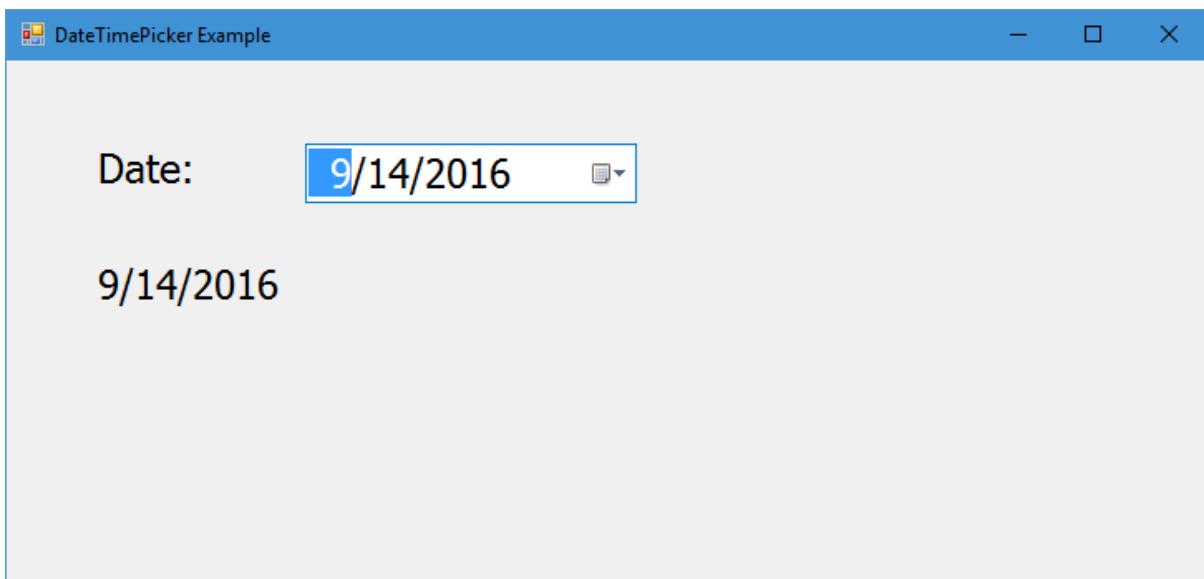
private void Datetimetype1_ValueChanged(object sender, EventArgs e)
{
    DateTime dt = datetimetype1.Value;
    label2.Text = dt.ToShortDateString();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Selected date appears in the label.

“System.Windows.Forms.MonthCalendar” class

- “MonthCalendar” is a class; “System.Windows.Forms” is a namespace.
- The “MonthCalendar” class / control is used to show a calendar and allow the user to select a date. It shows calendar directly (without dropdown button).

Steps for development of MonthCalendar:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `MonthCalendar referencevariable;`
- Create an object:
 - `referencevariable = new MonthCalendar();`

- Set properties:
 - *referencevariable.property = value;*
- Add event:
 - *referencevariable.event = method;*
- Add control to the form:
 - *this.Controls.Add(referencevariable);*

HARSHA

Properties of "MonthCalendar" class:

Sl. No	Property	Description
1	SelectionStart	Represents the currently selected starting date. <u>Syntax:</u> <code>referencevariable.SelectionStart = any date;</code>
2	SelectionEnd	Represents the currently selected ending date. <u>Syntax:</u> <code>referencevariable.SelectionEnd = any date;</code>
3	MaxSelectionCount	Represents the maximum no. of dates that can be selected in the control. <u>Default:</u> 7 <u>Syntax:</u> <code>referencevariable.MaxSelectionCount = any number;</code>
4	MinDate	Represents the minimum date that can be selected in the control. <u>Syntax:</u> <code>referencevariable.MinDate = any date;</code>
5	MaxDate	Represents the maximum date that can be selected in the control. <u>Syntax:</u> <code>referencevariable.MaxDate = any date;</code>
6	BoldedDates	Represents an array of dates that should appear in bold. <u>Syntax:</u> <code>referencevariable.BoldedDates = new DateTime[] { value1, value2, ... };</code>
7	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
8	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>

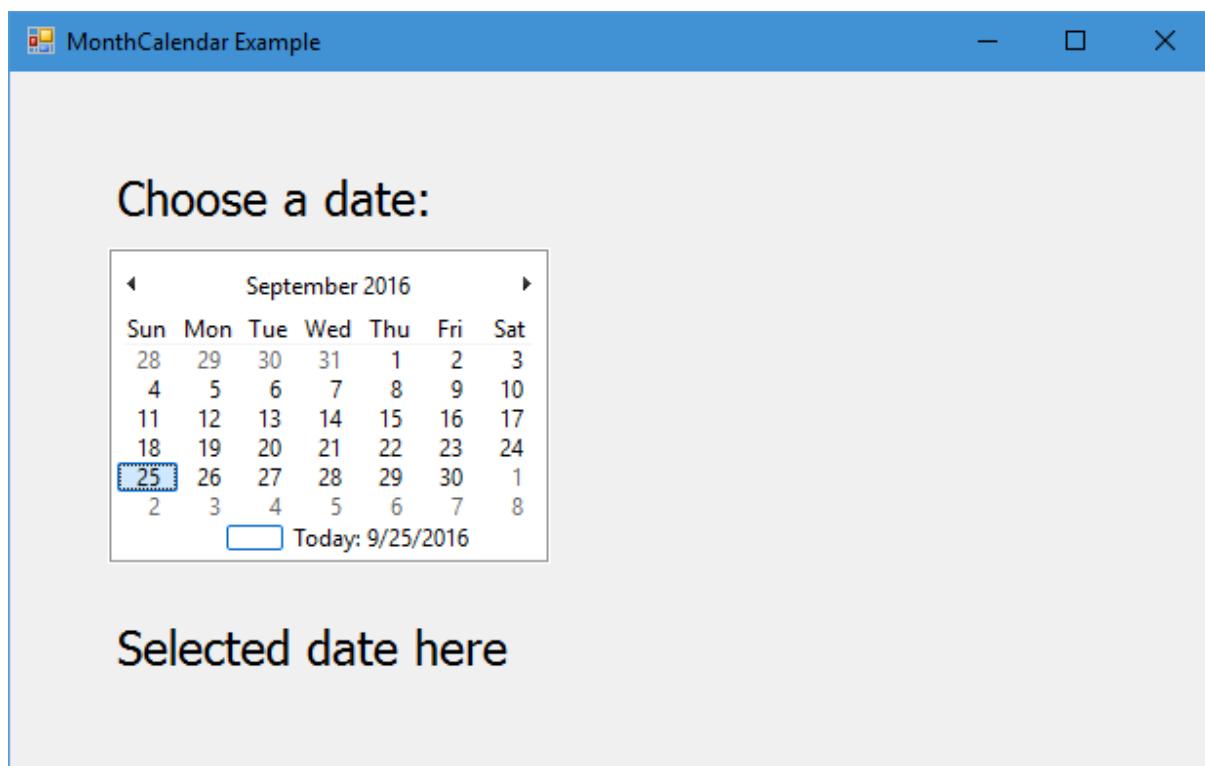
9	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code>
10	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code>
11	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
12	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>
13	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>
14	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <code>referencevariable.Enabled = true false;</code>
15	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code>

Events of "MonthCalendar" class:

Sl. No	Event	Description
1	DateSelected	Executes when the user selects a date in the control. <u>Syntax:</u> <i>referencevariable.DateSelected += methodname;</i>
2	Enter	Executes when the cursor enters into the control. <u>Syntax:</u> <i>referencevariable.Enter += methodname;</i>
3	Leave	Executes when the cursor leaves into the control. <u>Syntax:</u> <i>referencevariable.Leave += methodname;</i>
4	KeyPress	Executes when the user presses any key on the keyboard (before accepting the character into the control). <u>Syntax:</u> <i>pointername.KeyPress += methodname;</i>

“System.Windows.Forms.MonthCalendar” class – Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MonthCalendarExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MonthCalendarExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace MonthCalendarExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        MonthCalendar monthcalendar1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "MonthCalendar Example";
            this.Size = new Size(300, 200);
        }
    }
}
```

```

this.Font = new Font("Tahoma", 18);

/* creating label1 */
label1 = new Label();
label1.Text = "Choose a date: ";
label1.AutoSize = true;
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* creating label2 */
label2 = new Label();
label2.Text = "Selected date here";
label2.AutoSize = true;
label2.Location = new Point(50, 280);
this.Controls.Add(label2);

/* creating monthcalendar1 */
monthcalendar1 = new MonthCalendar();
monthcalendar1.Location = new Point(50, 90);
monthcalendar1.DateSelected += Monthcalendar1_DateSelected;
this.Controls.Add(monthcalendar1);

}

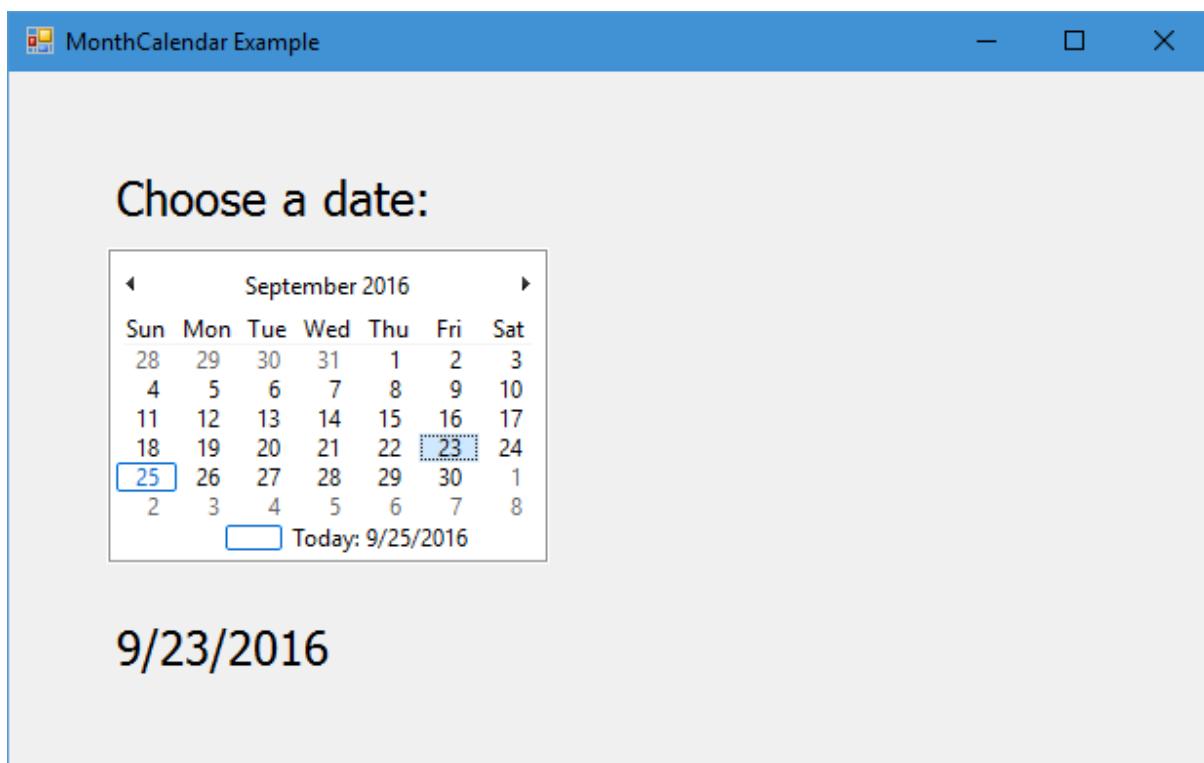
private void Monthcalendar1_DateSelected(object sender,
DateRangeEventArgs e)
{
    DateTime dt = monthcalendar1.SelectionStart;
    label2.Text = dt.ToShortDateString();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Selected date appears in the label.

“System.Windows.Forms.ToolTip” class

The “System.Windows.Forms.ToolTip” class

- “Tooltip” is a class; “System.Windows.Forms” is a namespace.
- The “ToolTip” class / control is used to display a tooltip message (help message) to the user when the user places the mouse pointer on a control.

Steps for development of ToolTip:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:

- `ToolTip referencevariable;`

- Create an object:
 - `referencevariable = new ToolTip();`

- Set properties:
 - `referencevariable.property = value;`

- Set tooltip
 - `referencevariable.SetToolTip(control name, "tooltip message");`

Properties of "Tooltip" class:

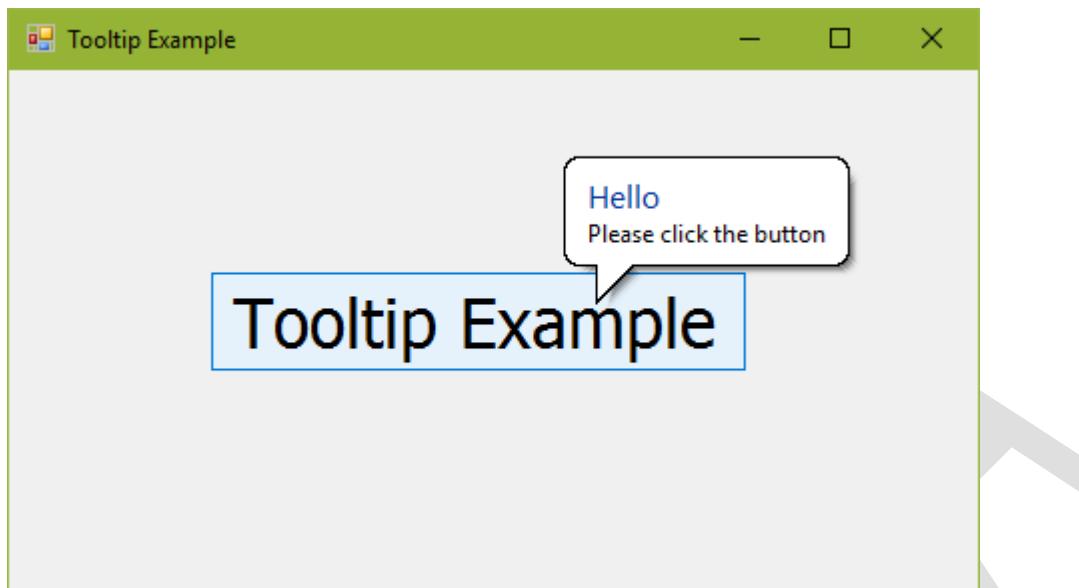
Sl. No	Property	Description
1	<code>IsBalloon</code>	<p><u>True</u>: The tooltip appears as a balloon.</p> <p><u>False</u>: The tooltip appears as a normal message.</p> <p><u>Syntax</u>: <code>referencevariable.IsBalloon = true false;</code></p>
2	<code>ToolTipTitle</code>	<p>Represents the title of the tooltip.</p> <p><u>Syntax</u>: <code>referencevariable.ToolTipTitle = "any text here";</code></p>

Methods of "ToolTip" class:

Sl. No	Method	Description
1	<code>SetToolTip</code>	<p>Sets tooltip message for a control.</p> <p><u>Syntax</u>: <code>referencevariable.SetToolTip(control name, "any text here");</code></p>

“System.Windows.Forms.ToolTip” class – Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TooltipExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TooltipExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace TooltipExample
{
    public partial class Form1 : Form
    {
        //create a reference variables
        Button button1;
        ToolTip tt;

        public Form1()
        {
            InitializeComponent();

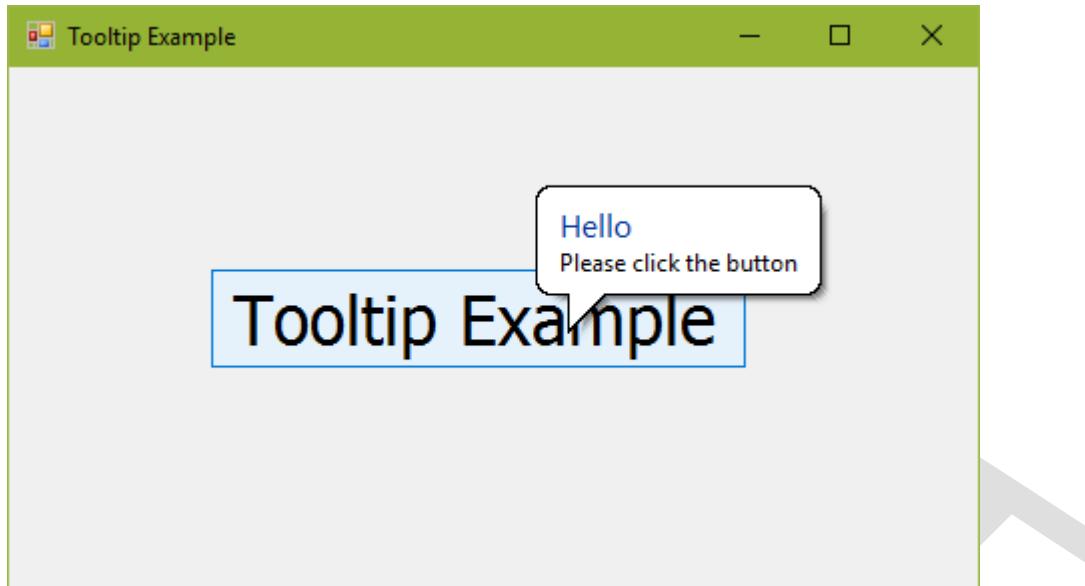
            //form properties
            this.Font = new Font("Tahoma", 25);
            this.Size = new Size(500, 300);
            this.Text = "Tooltip Example";

            //button1
            button1 = new Button();
            button1.Text = "Tooltip Example";
            button1.AutoSize = true;
            button1.Location = new Point(100, 100);
            this.Controls.Add(button1);

            //tooltip
            tt = new ToolTip();
            tt.IsBalloon = true;
            tt.ToolTipTitle = "Hello";
            tt.SetToolTip(button1, "Please click the button");
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

When you place the mouse pointer on the button, it shows the tooltip message.

"System.Windows.Forms.ErrorProvider" class

The "System.Windows.Forms.ErrorProvider" class

- "ErrorProvider" is a class; "System.Windows.Forms" is a namespace.
- The "ErrorProvider" class / control is used to perform validations in windows forms applications.
- "Validation" is a process of checking the form values whether those are correct or not.

Steps for development of ErrorProvider:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `ErrorProvider referencevariable;`

- Create an object:
 - `referencevariable = new ErrorProvider();`

- Set properties:
 - `referencevariable.property = value;`

- Set validation error message:
 - `referencevariable.SetError(control name, "error message");`

Properties of "ErrorProvider" class:

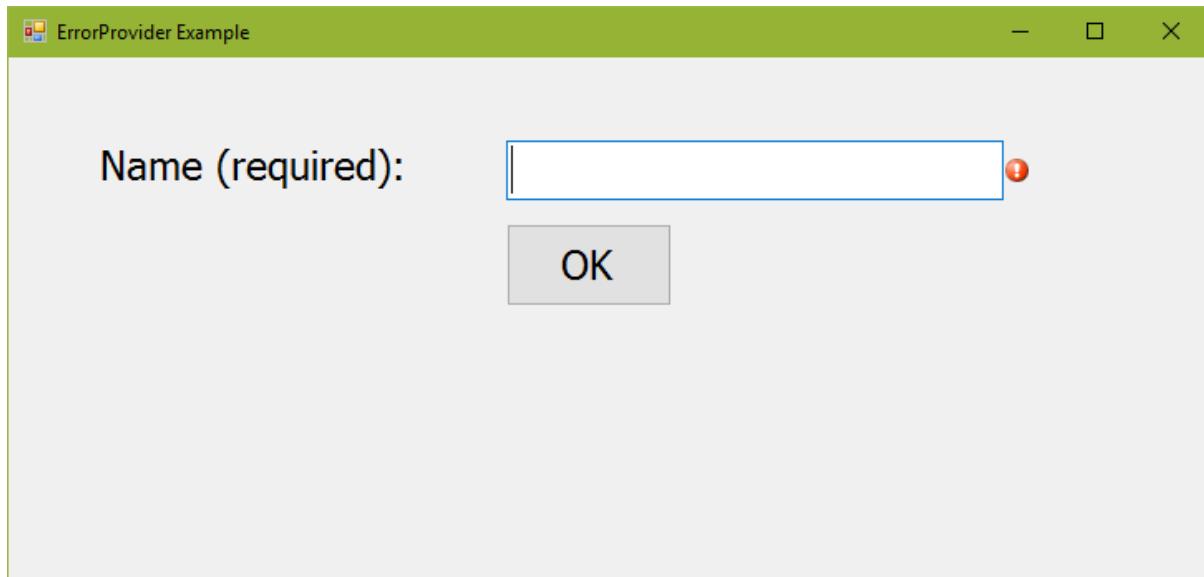
Sl. No	Property	Description
1	AlwaysBlink	<p><u>True:</u> The “!” marks blinks continuously.</p> <p><u>False:</u> The “!” marks blinks only few times.</p> <p><u>Syntax:</u> <code>referencevariable.AlwaysBlink = true false;</code></p>

Methods of "ErrorProvider" class:

Sl. No	Method	Description
1	SetError	<p>Sets an error message to the control</p> <p><u>Syntax:</u> <code>referencevariable.SetError(control name, "any error here");</code></p>

“System.Windows.Forms.ErrorProvider” class – Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ErrorProviderExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ErrorProviderExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
```

```
namespace ErrorProviderExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1;
        TextBox textbox1;
        Button button1;
        ErrorProvider errorprovider1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "ErrorProvider Example";
            this.Size = new Size(350, 180);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Name (required): ";
            label1.AutoSize = true;
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* creating textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(300, 30);
            textbox1.Location = new Point(300, 50);
            this.Controls.Add(textbox1);

            /* creating button1 */
            button1 = new Button();
            button1.Text = "OK";
            button1.Size = new Size(100, 50);
            button1.Location = new Point(300, 100);
            button1.Click += Button1_Click;
            this.Controls.Add(button1);

            /* creating errorprovider1 */
        }
    }
}
```

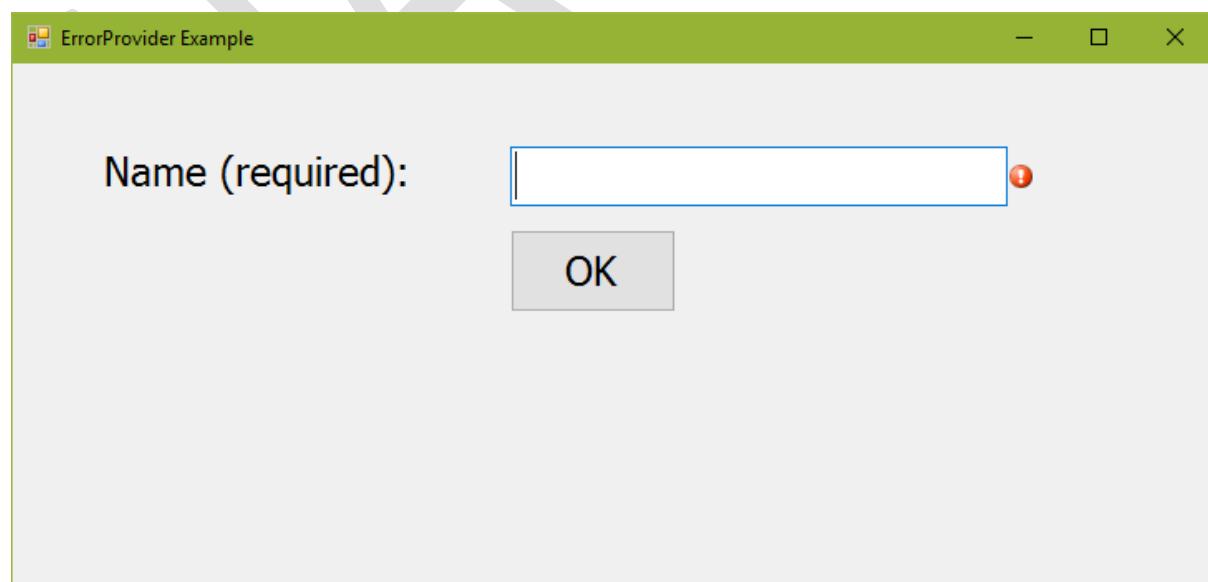
```
errorprovider1 = new ErrorProvider();
errorprovider1.BlinkStyle = ErrorBlinkStyle.AlwaysBlink;
}

private void Button1_Click(object sender, EventArgs e)
{
    if (textbox1.Text == "")
    {
        errorprovider1.SetError(textbox1, "Name is mandatory");
        textbox1.Focus();
    }
    else
    {
        errorprovider1.SetError(textbox1, "");
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you click on OK button without entering the name, it shows blinking “!” symbol. When you place the mouse pointer on “!” icon it shows error message.

“System.Text.RegularExpressions.Regex” class

The “System.Text.RegularExpressions.Regex” class

- “Regex” is a class; “System.Text.RegularExpressions” is a namespace.
- The “Regex” class is used to check pattern (format) of a value.
- It is useful in validations. Ex: Alphabets only allowed, Numbers only allowed, email address only allowed, Mobile number only allowed etc.

Steps for development of Regex:

- Import the namespace:
 - `using System.Text.RegularExpressions;`
- Create a reference variable:
 - `Regex referencevariable;`
- Create an object:
 - `referencevariable = new Regex(“regular expression here”);`
- Check for the match:
 - `referencevariable.Match(value);`

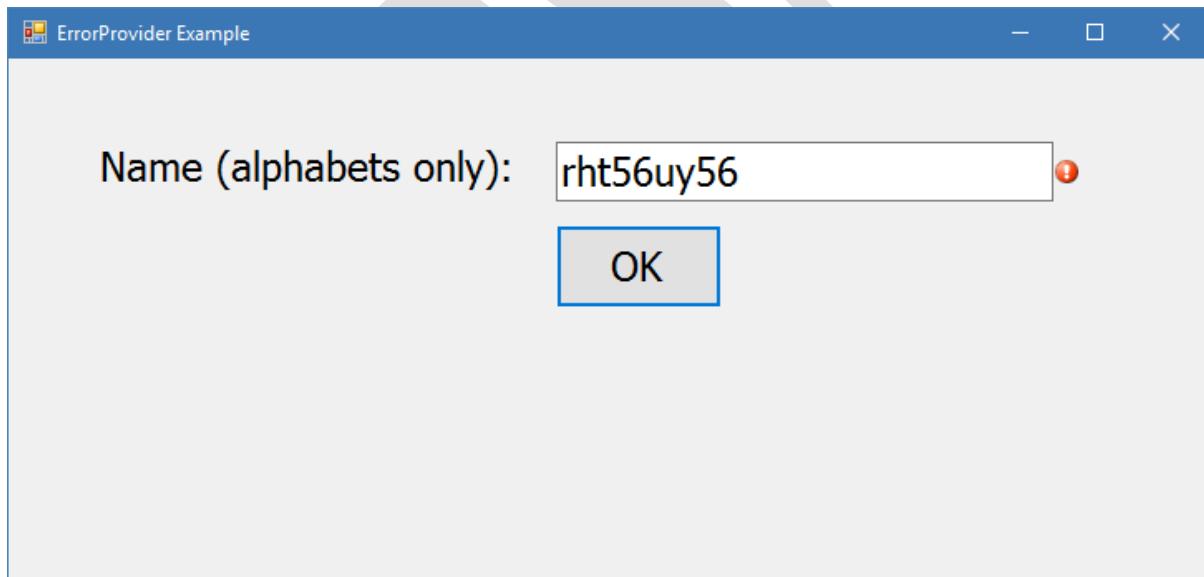
Constructors of “Regex” class:

Sl. No	Constructor	Description

1	Regex	Initializes regular expression. <u>Syntax:</u> new Regex("regular expression here");
---	-------	---

Methods of "Regex" class:

Sl. No	Method	Description
1	Match	Checks whether the value is matching with given expression or not. <u>Syntax:</u> referencevariable.Match(value);

"System.Text.RegularExpressions. Regex" class – Example**Expected Output****Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.

- Type the project name as “RegularExpressionsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RegularExpressionsExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Text.RegularExpressions;

namespace RegularExpressionsExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1;
        TextBox textbox1;
        Button button1;
        ErrorProvider errorprovider1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "ErrorProvider Example";
            this.Size = new Size(350, 180);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Name (alphabets only): ";
        }
    }
}
```

```
label1.AutoSize = true;
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* creating textBox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(300, 30);
textBox1.Location = new Point(330, 50);
this.Controls.Add(textBox1);

/* creating button1 */
button1 = new Button();
button1.Text = "OK";
button1.Size = new Size(100, 50);
button1.Location = new Point(330, 100);
button1.Click += Button1_Click;
this.Controls.Add(button1);

/* creating errorprovider1 */
errorprovider1 = new ErrorProvider();
errorprovider1.BlinkStyle = ErrorBlinkStyle.AlwaysBlink;
}

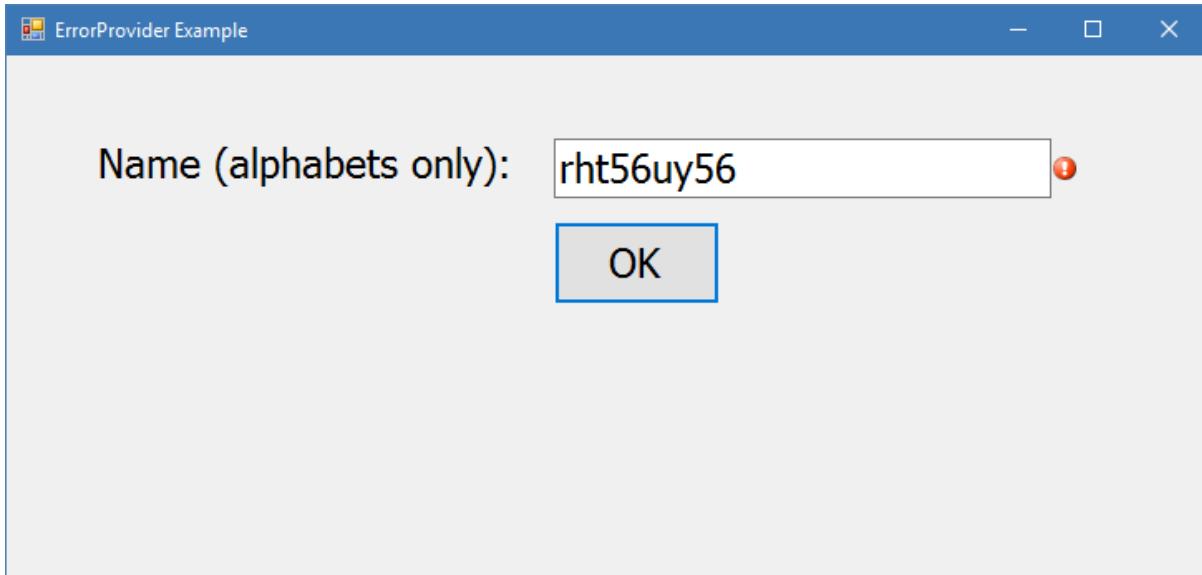
private void Button1_Click(object sender, EventArgs e)
{
    Regex regex = new Regex(@"^*[a-zA-Z]*$");
    Match m = regex.Match(textBox1.Text);
    if (m.Success == false)
    {
        errorprovider1.SetError(textBox1, "Alphabets only allowed");
    }
    else
    {
        errorprovider1.SetError(textBox1, "");
    }
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you type alphabets and numbers and then click on OK button, it shows blinking “!” symbol.

When you place the mouse pointer on “!” icon it shows error message. If you enter alphabets only, it hides the “!” symbol. That means it is accepted.

“System.Windows.Forms. MaskedTextBox” class

The “System.Windows.Forms.MaskedTextBox” class

- “MaskedTextBox” is a class; “System.Windows.Forms” is a namespace.
- The “MaskedTextBox” class / control is used to accept a value from the user, based on a specific pattern (__/__/____). Ex: Date

Steps for development of MaskedTextBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `MaskedTextBox referencevariable;`
- Create an object:
 - `referencevariable = new MaskedTextBox();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “MaskedTextBox” class:

Sl. No	Property	Description
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Text = "text here";</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
5	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
6	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
7	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>

8	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
9	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
10	TextAlign	Represents the alignment of the text in the control. <u>Options:</u> Left Center Right <u>Syntax:</u> <i>referencevariable.TextAlign = System.Windows.Forms.HorizontalAlignment.Left;</i>
11	Readonly	<u>True:</u> The user can't modify the value of textbox. <u>False:</u> The user can modify the value of textbox. <u>Syntax:</u> <i>referencevariable.Readonly = true false;</i>
12	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>pointername.TabIndex = any number;</i>
13	Mask	Represents mask (format) of the control. <u>Syntax:</u> <i>referencevariable.Mask = "any mask";</i>
14	PromptChar	Represents the character, which should be shown to the user in place of a character. <u>Syntax:</u> <i>referencevariable.PromptChar = 'any character';</i>

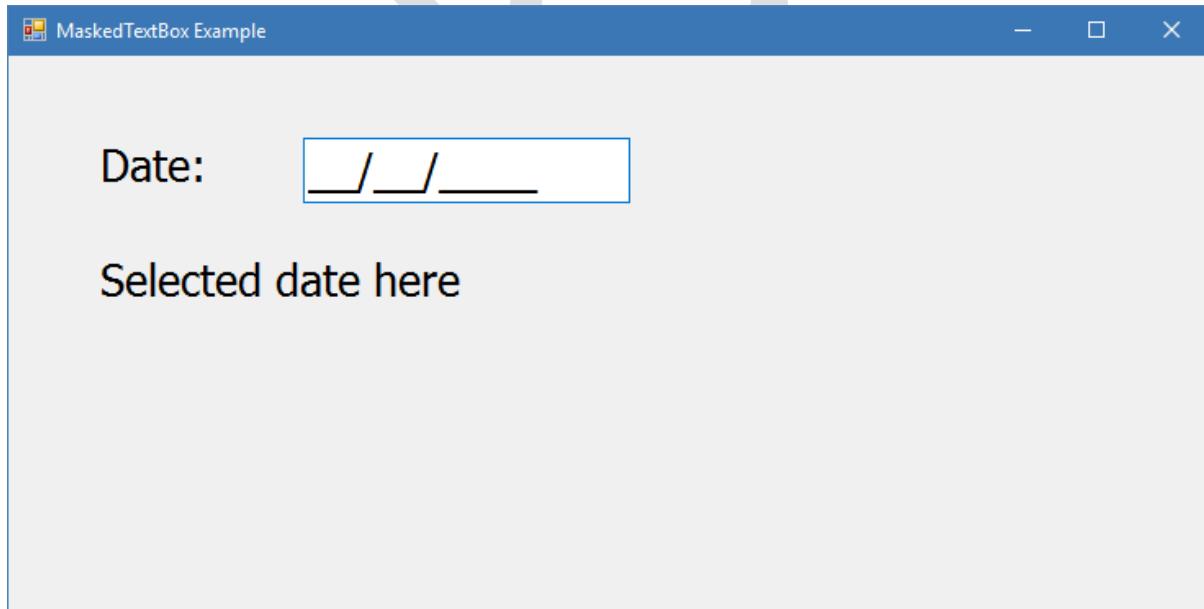
Events of "MaskedTextBox" class:

Sl. No	Event	Description

1	TextChanged	Executes when the user types a character in the control. <u>Syntax:</u> <code>referencevariable.TextChanged += methodname;</code>
2	Enter	Executes when the cursor enters into the control. <u>Syntax:</u> <code>referencevariable.Enter += methodname;</code>
3	Leave	Executes when the cursor leaves from the control. <u>Syntax:</u> <code>referencevariable.Leave += methodname;</code>
4	Click	Executes when the user clicks on the control. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>

“System.Windows.Forms. MaskedTextBox” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MaskedTextBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MaskedTextBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace MaskedTextBoxExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        MaskedTextBox maskedtextbox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "MaskedTextBox Example";
            this.Font = new System.Drawing.Font("Tahoma", 20);
            this.Size = new System.Drawing.Size(750, 380);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Date: ";
            label1.AutoSize = true;
```

```
label1.Location = new System.Drawing.Point(50, 50);
this.Controls.Add(label1);

/* creating label2 */
label2 = new Label();
label2.Text = "Selected date here";
label2.AutoSize = true;
label2.Location = new System.Drawing.Point(50, 120);
this.Controls.Add(label2);

/* creating maskedtextbox1 */
maskedtextbox1 = new MaskedTextBox();
maskedtextbox1.Size = new System.Drawing.Size(200, 30);
maskedtextbox1.Location = new System.Drawing.Point(180, 50);
maskedtextbox1.Mask = "00/00/0000";
maskedtextbox1.TextChanged += Maskedtextbox1_TextChanged;
maskedtextbox1.PromptChar = '_';
this.Controls.Add(maskedtextbox1);

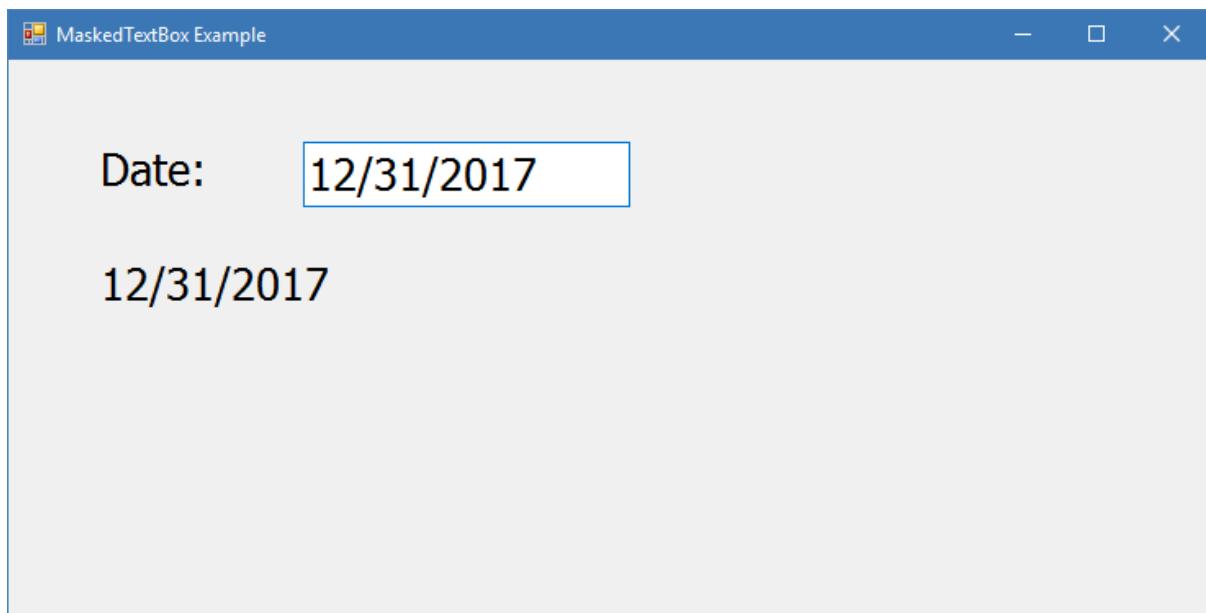
}

private void Maskedtextbox1_TextChanged(object sender, EventArgs e)
{
    label2.Text = maskedtextbox1.Text;
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows mask (___/___/_____) in the textbox.

"System.Windows.Forms.CheckBox" class

The "System.Windows.Forms.CheckBox" class

- "CheckBox" is a class; "System.Windows.Forms" is a namespace.
- The "CheckBox" class / control is used to accept Yes / No type of option to the user. Ex:
Accept license agreement

Steps for development of CheckBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `CheckBox referencevariable;`
- Create an object:

- `referencevariable = new CheckBox();`

- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event += method;`

- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "CheckBox" class:

Sl. No	Property	Description
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Text = "text here";</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>

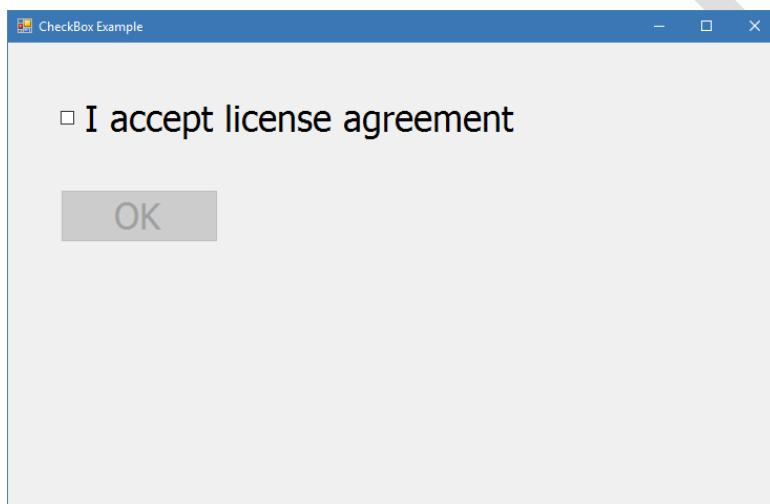
5	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
6	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
7	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
8	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>
9	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
10	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
11	Checked	<p>Represents the current status of the checkbox.</p> <p><u>True:</u> The checkbox is currently checked.</p> <p><u>False:</u> The checkbox is currently unchecked.</p> <p><u>Syntax:</u> <code>referencevariable.Checked = true false;</code></p>

Events of "CheckBox" class:

Sl. No	Event	Description
1	CheckedChanged	Executes when the user checks / unchecks the checkbox. <u>Syntax:</u> <code>referencevariable.CheckedChanged += methodname;</code>

“System.Windows.Forms. CheckBox” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “CheckBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CheckBoxExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace CheckBoxExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        CheckBox checkbox1;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "CheckBox Example";
            this.Font = new Font("Tahoma", 25);
            this.Size = new Size(750, 480);

            /* creating label1 */
            checkbox1 = new CheckBox();
            checkbox1.Text = "I accept license agreement";
            checkbox1.AutoSize = true;
            checkbox1.Location = new Point(50, 50);
            checkbox1.CheckedChanged += Checkbox1_CheckedChanged;
            this.Controls.Add(checkbox1);

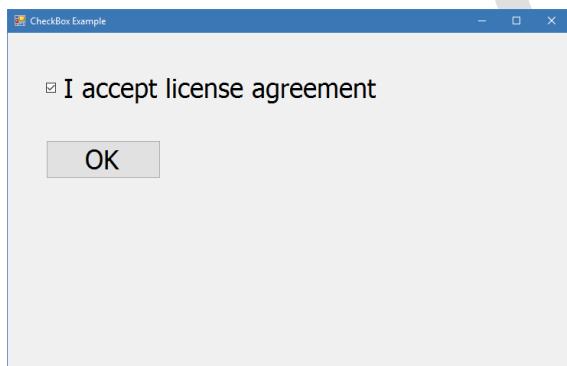
            /* creating button1 */
            button1 = new Button();
            button1.Text = "OK";
            button1.Size = new Size(150, 50);
            button1.Location = new Point(50, 140);
            button1.Enabled = false;
            this.Controls.Add(button1);
        }
    }
}
```

```
private void Checkbox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkbox1.Checked == true)
        button1.Enabled = true;
    else
        button1.Enabled = false;
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you check the checkbox, the button will be enabled.

When you uncheck the checkbox, the button will be disabled.

“System.Windows.Forms. RadioButton” class

The “System.Windows.Forms.RadioButton” class

- “RadioButton” is a class; “System.Windows.Forms” is a namespace.

- The “RadioButton” class / control is used to display a set of options to the user and allow the user to select any one of them. Ex: Male, Female.

Steps for development of RadioButton:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `RadioButton referencevariable;`
- Create an object:
 - `referencevariable = new RadioButton();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “RadioButton” class:

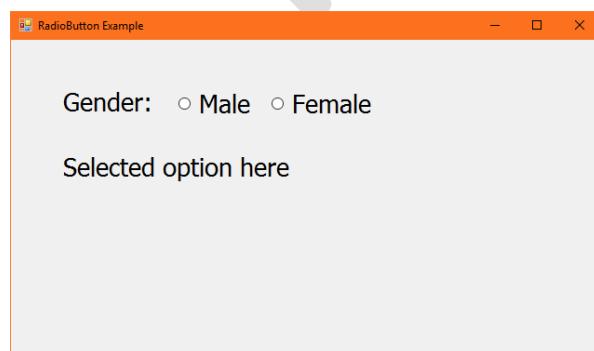
Sl. No	Property	Description
1	<code>Text</code>	Represents text of the control. <u>Syntax:</u> <code>referencevariable.Text = "text here";</code>

2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
5	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
6	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
7	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
8	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

9	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
10	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>
11	Checked	Represents the current status of the radio button. <u>True:</u> The radio button is currently checked. <u>False:</u> The radio button is currently unchecked. <u>Syntax:</u> <i>referencevariable.Checked = true false;</i>

Events of "RadioButton" class:

Sl. No	Event	Description
1	CheckedChanged	Executes when the user checks / unchecks the radio button. <u>Syntax:</u> <i>referencevariable.CheckedChanged += methodname;</i>

"System.Windows.Forms. RadioButton" class - Example**Expected Output****Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “RadioButtonExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RadioButtonExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace RadioButtonExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        RadioButton radiobutton1, radiobutton2;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "RadioButton Example";
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 380);
        }
    }
}
```

```

/* creating label1 */
label1 = new Label();
label1.Text = "Gender:";
label1.AutoSize = true;
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* creating label2 */
label2 = new Label();
label2.Text = "Selected option here";
label2.AutoSize = true;
label2.Location = new Point(50, 120);
this.Controls.Add(label2);

/* creating radiobutton1 */
radiobutton1 = new RadioButton();
radiobutton1.Text = "Male";
radiobutton1.AutoSize = true;
radiobutton1.Location = new Point(180, 50);
radiobutton1.CheckedChanged += Radiobutton1_CheckedChanged;
this.Controls.Add(radiobutton1);

/* creating radiobutton2 */
radiobutton2 = new RadioButton();
radiobutton2.Text = "Female";
radiobutton2.AutoSize = true;
radiobutton2.Location = new Point(280, 50);
radiobutton2.CheckedChanged += Radiobutton2_CheckedChanged;
this.Controls.Add(radiobutton2);

}

private void Radiobutton1_CheckedChanged(object sender, EventArgs e)
{
    if (radiobutton1.Checked == true)
        label2.Text = "Male selected";
}

```

```

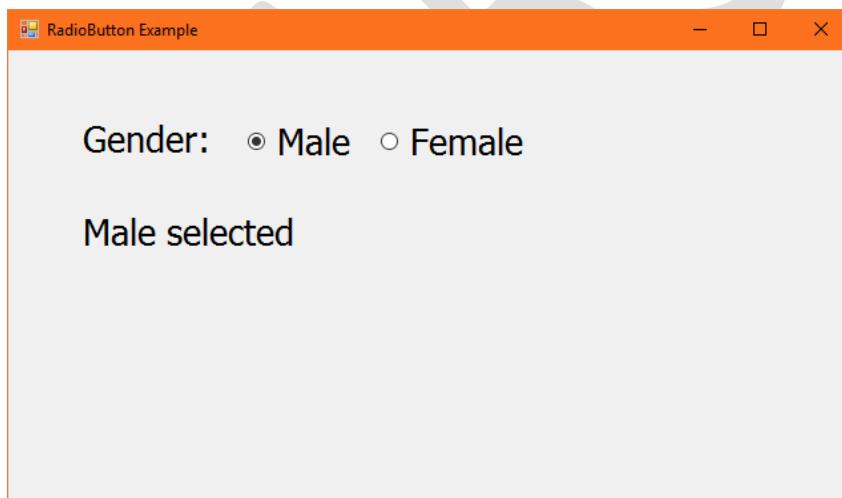
    else
        label2.Text = "Female selected";
    }

private void RadioButton2_CheckedChanged(object sender, EventArgs
e)
{
    if (radioButton1.Checked == true)
        label2.Text = "Male selected";
    else
        label2.Text = "Female selected";
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

When you click on “Male” radio button, it shows “Male selected”.

When you click on “Female” radio button, it shows “Female selected”.

“System.Windows.Forms.ComboBox” class

The “System.Windows.Forms.ComboBox” class

- “ComboBox” is a class; “System.Windows.Forms” is a namespace.
- The “ComboBox” class / control is used to display a set of options to the user and allow the user to select any one of them. Ex: Countries: India, UK, USA etc.
- It occupies less space than radio button.

Steps for development of ComboBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `ComboBox referencevariable;`
- Create an object:
 - `referencevariable = new ComboBox();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “ComboBox” class:

Sl. No	Property	Description
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Text = "text here";</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
5	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
6	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
7	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>

8	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
9	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
10	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>
11	Items	Represents the collection of items, that should be appear in the combo box. <u>Syntax:</u> <i>referencevariable.Items.AddRange(string array here);</i>
12	DropDownStyle	Specifies type of selection in the combo box. <u>Options:</u> DropDown DropDownList. <u>Syntax:</u> <i>referencevariable.DropDownStyle = System.Windows.Forms.ComboBoxStyle.option here;</i>
13	SelectedItem	Represents the currently selected item in "System.Object" data type. <u>Syntax:</u> <i>referencevariable.SelectedItem = value;</i>
14	SelectedIndex	Represents the index of currently selected item in the combo box. Index starts from 0. If no item is selected, it returns -1. <u>Syntax:</u> <i>referencevariable.SelectedIndex= any number;</i>

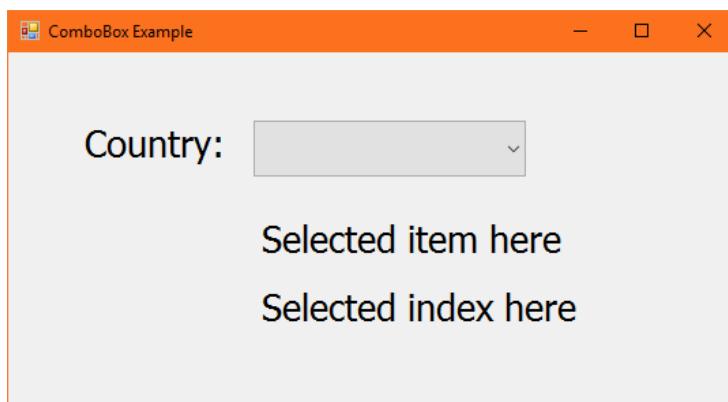
Events of "ComboBox" class:

Sl. No	Event	Description

1	SelectedIndexChanged	Executes when the user selects an item in the combo box. <u>Syntax:</u> <i>referencevariable.SelectedIndexChanged += methodname;</i>
---	----------------------	---

“System.Windows.Forms.ComboBox” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ComboBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ComboBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

using System;

```
using System.Windows.Forms;
using System.Drawing;

namespace ComboBoxExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2, label3;
        ComboBox combobox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "ComboBox Example";
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Country: ";
            label1.AutoSize = true;
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* creating label2 */
            label2 = new Label();
            label2.Text = "Selected item here";
            label2.AutoSize = true;
            label2.Location = new Point(180, 120);
            this.Controls.Add(label2);

            /* creating label3 */
            label3 = new Label();
            label3.Text = "Selected index here";
        }
    }
}
```

```

label3.AutoSize = true;
label3.Location = new Point(180, 170);
this.Controls.Add(label3);
/* creating combobox1 */
combobox1 = new ComboBox();
combobox1.Size = new Size(200, 40);
combobox1.Location = new Point(180, 50);
combobox1.DropDownStyle = ComboBoxStyle.DropDownList;
string[] countries = new string[] { "India", "China", "UK", "USA", "Japan" };
combobox1.Items.AddRange(countries);
combobox1.SelectedIndexChanged +=
Combobox1_SelectedIndexChanged;
this.Controls.Add(combobox1);
}

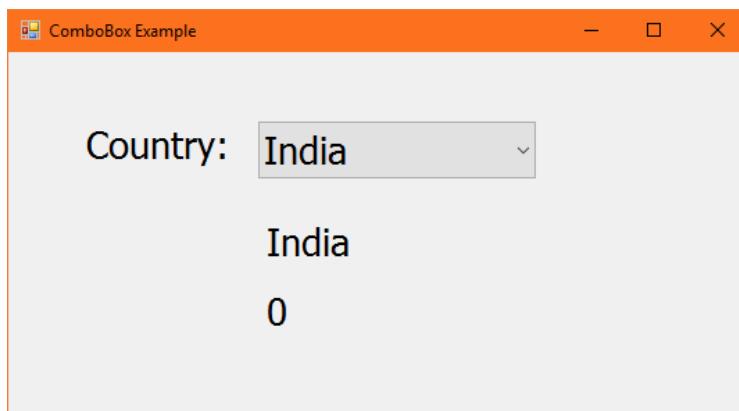
private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    /* get selected item */
    object obj = combobox1.SelectedItem;
    string s = Convert.ToString(obj);
    label2.Text = s;
    /* get selected index */
    int n = combobox1.SelectedIndex;
    label3.Text = Convert.ToString(n);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

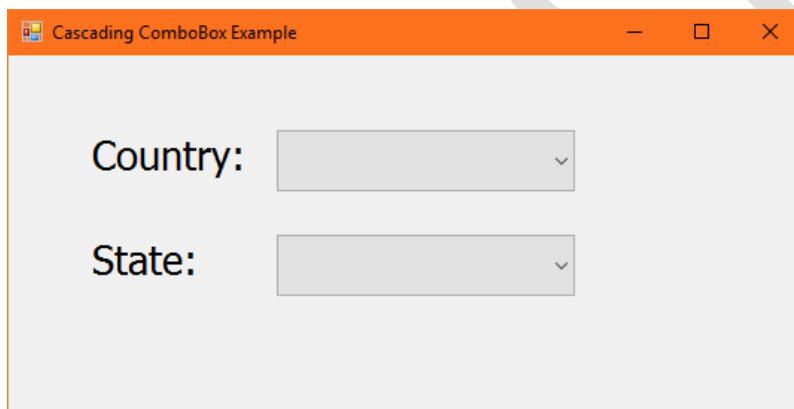
Output



When you select a country, it shows its name and index.

"Cascading ComboBox" - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".
- Select "Windows Forms Application".
- Type the project name as "CascadingComboBoxExample".
- Type the location as "C:\CSharp".
- Type the solution name as "CascadingComboBoxExample".

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;

namespace CascadingComboBoxExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        ComboBox combobox1, combobox2;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "Cascading ComboBox Example";
            this.Font = new System.Drawing.Font("Tahoma", 20);
            this.Size = new System.Drawing.Size(550, 280);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Country: ";
            label1.AutoSize = true;
            label1.Location = new System.Drawing.Point(50, 50);
            this.Controls.Add(label1);

            /* creating label2 */
            label2 = new Label();
            label2.Text = "State: ";
            label2.AutoSize = true;
```

```

label2.Location = new System.Drawing.Point(50, 120);
this.Controls.Add(label2);

/* creating combobox1 */
combobox1 = new ComboBox();
combobox1.Size = new System.Drawing.Size(200, 40);
combobox1.Location = new System.Drawing.Point(180, 50);
combobox1.DropDownStyle = ComboBoxStyle.DropDownList;
combobox1.Items.AddRange(new string[] { "India", "UK", "USA" });
combobox1.SelectedIndexChanged +=
Combobox1_SelectedIndexChanged;
this.Controls.Add(combobox1);

/* creating combobox2 */
combobox2 = new ComboBox();
combobox2.Size = new System.Drawing.Size(200, 40);
combobox2.Location = new System.Drawing.Point(180, 120);
combobox2.DropDownStyle = ComboBoxStyle.DropDownList;
this.Controls.Add(combobox2);
}

private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    int n = combobox1.SelectedIndex;
    if (n == 0)
    {
        combobox2.Items.Clear();
        combobox2.Items.AddRange(new string[] { "A", "B", "C" });
    }
    else if (n == 1)
    {
        combobox2.Items.Clear();
        combobox2.Items.AddRange(new string[] { "D", "E", "F" });
    }
    else if (n == 2)
    {
}

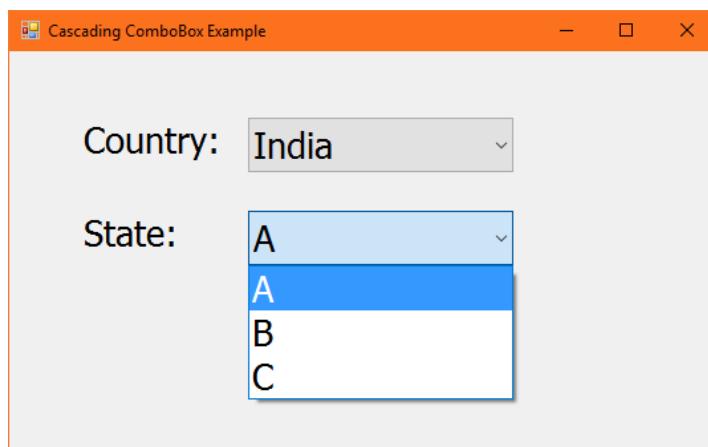
```

```
    combobox2.Items.Clear();
    combobox2.Items.AddRange(new string[] { "G", "H", "I" });
}
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you select a country in the first combo box, it shows its list of states in the second combo box.

“System.Windows.Forms. ListBox” class

The “**System.Windows.Forms.ListBox**” class

- “LisBox” is a class; “System.Windows.Forms” is a namespace.
 - The “ListBox” class / control displays a set of options to the user and allow the user to select one or more them. Ex: Countries: India, UK, USA etc.
 - It occupies more space than combo box.

Steps for development of ListBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `ListBox referencevariable,`
- Create an object:
 - `referencevariable = new ListBox();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "ListBox" class:

Sl. No	Property	Description
1	Text	Represents text of the control. <u>Syntax:</u> <code><i>referencevariable.Text</i> = "text here";</code>

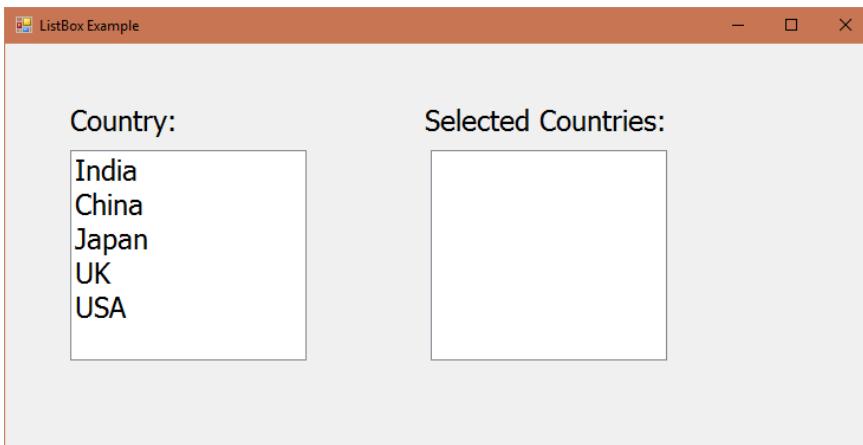
2	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
4	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code>
5	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code>
6	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
7	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>
8	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>

9	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
10	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>
11	Items	Represents the collection of items, that should be appear in the listbox. <u>Syntax:</u> <i>referencevariable.Items.AddRange(string array here);</i>
12	SelectedItems	Represents the collection of currently selected items. <u>Syntax:</u> <i>referencevariable.SelectedItems</i>

Events of "ListBox" class:

Sl. No	Event	Description
1	SelectedIndexChanged	Executes when the user selects an item in the listbox. <u>Syntax:</u> <i>referencevariable.SelectedIndexChanged += methodname;</i>

"System.Windows.Forms.ListBox" class - Example**Expected Output**



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ListBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ListBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```

using System;
using System.Windows.Forms;
using System.Drawing;

namespace ListBoxExample
{
    public partial class Form1 : Form
    {
        //create reference variables
    }
}

```

```
Label label1, label2;
ListBox listBox1, listBox2;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Text = "ListBox Example";
    this.Font = new Font("Tahoma", 18);
    this.Size = new Size(750, 380);
    /* creating label1 */
    label1 = new Label();
    label1.Text = "Country: ";
    label1.AutoSize = true;
    label1.Location = new Point(50, 50);
    this.Controls.Add(label1);
    /* creating label2 */
    label2 = new Label();
    label2.Text = "Selected Countries:";
    label2.AutoSize = true;
    label2.Location = new Point(350, 50);
    this.Controls.Add(label2);
    /* creating listBox1 */
    listBox1 = new ListBox();
    listBox1.Size = new Size(200, 200);
    listBox1.Location = new Point(55, 90);
    listBox1.SelectionMode = SelectionMode.MultiSimple;
    listBox1.Items.AddRange(new string[] { "India", "China", "Japan", "UK",
    "USA" });
    listBox1.SelectedIndexChanged += Listbox1_SelectedIndexChanged;
    this.Controls.Add(listBox1);
    /* creating listBox2 */
    listBox2 = new ListBox();
    listBox2.Size = new Size(200, 200);
    listBox2.Location = new Point(360, 90);
    this.Controls.Add(listBox2);
```

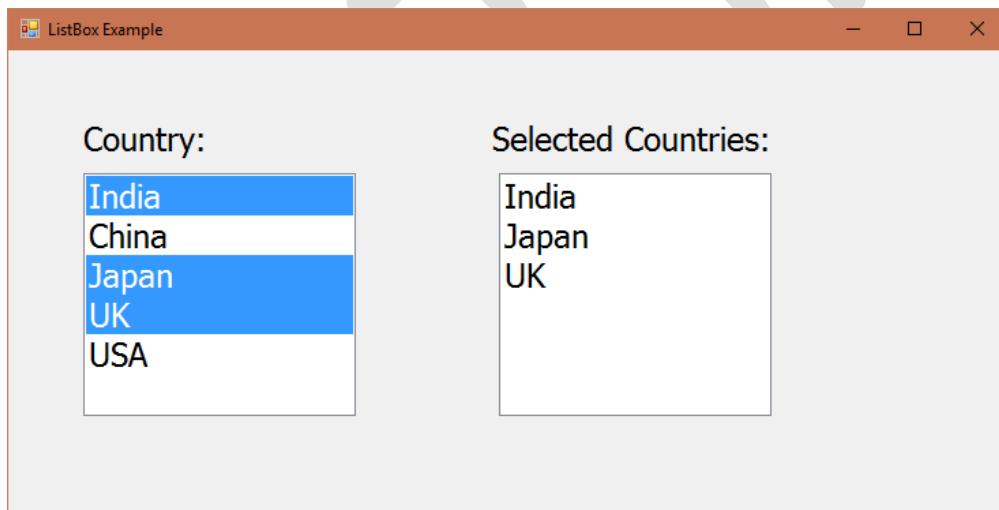
```

        }
    private void Listbox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        listbox2.Items.Clear();
        for (int i = 0; i < listbox1.SelectedItems.Count; i++)
        {
            listbox2.Items.Add(listbox1.SelectedItems[i]);
        }
    }
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

When you select any country names in the first listbox, only the selected names will appear in the second listbox.

"System.Windows.Forms.CheckedListBox" class

The "System.Windows.Forms.CheckedListBox" class

- “CheckedListbox” is a class; “System.Windows.Forms” is a namespace.
- The “CheckedListBox” class / control is used to display a set of options to the user and allow the user to select one or more them. Ex: Countries: India, UK, USA etc. It shows checkbox for each item.

Steps for development of CheckedListBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `CheckedListBox referencevariable;`
- Create an object:
 - `referencevariable = new CheckedListBox();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “CheckedListBox” class:

Sl. No	Property	Description

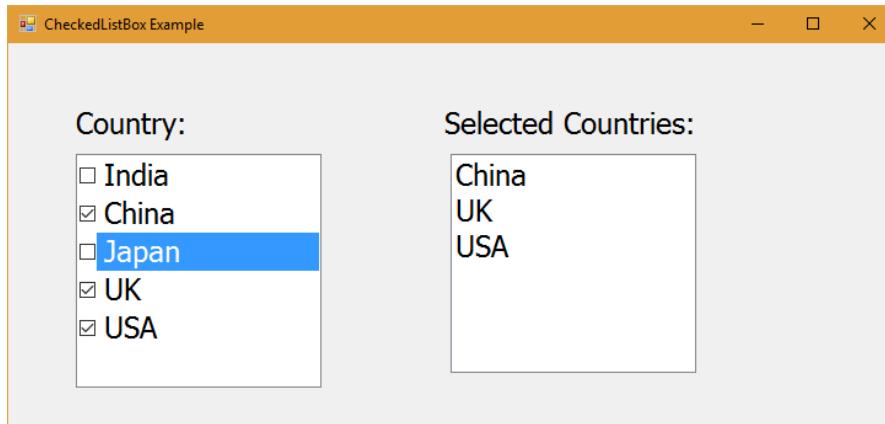
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Text = "text here";</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
5	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
6	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
7	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
8	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

9	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
10	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
11	Items	<p>Represents the collection of items, that should appear in the checked list box.</p> <p><u>Syntax:</u> <code>referencevariable.Items.AddRange(string array here);</code></p>
12	CheckedItems	<p>Represents the collection of currently checked items.</p> <p><u>Syntax:</u> <code>referencevariable.CheckedItems</code></p>
13	CheckOnClick	<p><u>True:</u> The checkbox will be checked when the user clicks on the item.</p> <p><u>False:</u> The checkbox will be checked when the user only clicks on the actual checkbox (not item).</p> <p><u>Syntax:</u> <code>referencevariable.CheckOnClick = true false;</code></p>

Events of "CheckedListBox" class:

Sl. No	Event	Description
1	SelectedIndexChanged	<p>Executes when the user checks an item in the checked listbox.</p> <p><u>Syntax:</u> <code>referencevariable.SelectedIndexChanged += methodname;</code></p>

"System.Windows.Forms.CheckedListBox" class – Example

Expected Output**Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “CheckedListBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CheckedListBoxExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```

using System;
using System.Windows.Forms;
using System.Drawing;

namespace CheckedListBoxExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        ListBox listBox1;
    }
}

```

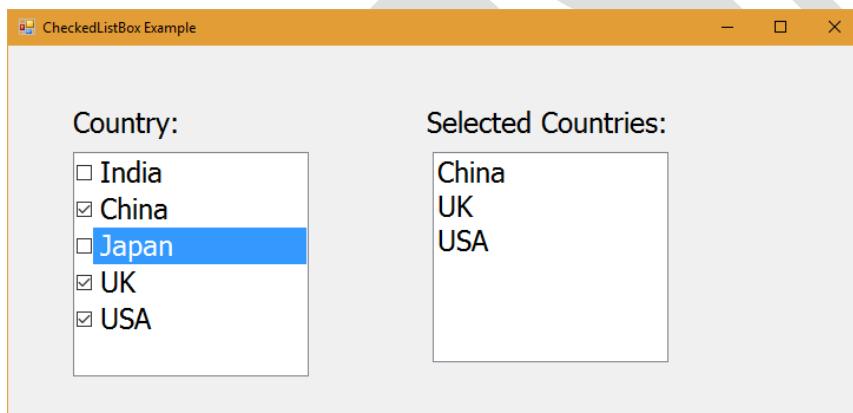
```
CheckedListBox checkedlistbox1;
public Form1()
{
    InitializeComponent();
    /* form properties */
    this.Text = "CheckedListBox Example";
    this.Size = new Size(350, 180);
    this.Font = new Font("Tahoma", 18);
    /* creating label1 */
    label1 = new Label();
    label1.Text = "Country: ";
    label1.AutoSize = true;
    label1.Location = new Point(50, 50);
    this.Controls.Add(label1);
    /* creating label2 */
    label2 = new Label();
    label2.Text = "Selected Countries:";
    label2.AutoSize = true;
    label2.Location = new Point(350, 50);
    this.Controls.Add(label2);
    /* creating checkedlistbox1 */
    checkedlistbox1 = new CheckedListBox();
    checkedlistbox1.Size = new Size(200, 200);
    checkedlistbox1.Location = new Point(55, 90);
    checkedlistbox1.Items.AddRange(new string[] { "India", "China",
    "Japan", "UK", "USA" });
    checkedlistbox1.CheckOnClick = true;
    checkedlistbox1.SelectedIndexChanged +=
    Checkedlistbox1_SelectedIndexChanged;
    this.Controls.Add(checkedlistbox1);
    /* creating listbox1 */
    listbox1 = new ListBox();
    listbox1.Size = new Size(200, 200);
    listbox1.Location = new Point(360, 90);
    this.Controls.Add(listbox1);
}
```

```
private void Checkedlistbox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    listbox1.Items.Clear();
    for (int i = 0; i < checkedlistbox1.CheckedItems.Count; i++)
    {
        listbox1.Items.Add(checkedlistbox1.CheckedItems[i]);
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



When you select any country names in the first checked listbox, only the selected names will appear in the second listbox.

“System.Windows.Forms.TreeView” class

- “TreeView” is a class; “System.Windows.Forms” is a namespace.

- The “TreeView” class / control is used to display a set of options in tree structure (parent and child structure). Ex: Folders and files.
- Each option (item) is called as “Node”. A node can have ‘n’ no. of child nodes.

Steps for development of TreeView:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `TreeView referencevariable;`
- Create an object:
 - `referencevariable = new TreeView();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “TreeView” class:

Sl. No	Property	Description

1	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
2	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
3	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
4	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
5	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
6	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
7	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

8	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
8	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
9	Nodes	<p>Represents the collection of nodes, that should appear in the tree view.</p> <p><u>Syntax:</u> <code>referencevariable.Nodes.AddRange(node array here);</code></p> <p><u>Syntax:</u> <code>referencevariable.Nodes[index];</code></p>
10	Checkboxes	<p><u>True:</u> Checkboxes will be shown for every node.</p> <p><u>False:</u> Checkboxes will not be shown for every node.</p> <p><u>Syntax:</u> <code>referencevariable.CheckBoxes = true false;</code></p>

Events of "TreeView" class:

Sl. No	Event	Description
1	AfterCheck	<p>Executes when the user checks a node in the treeview.</p> <p><u>Syntax:</u> <code>referencevariable.AfterCheck += methodname;</code></p>

Properties of "TreeNode" class:

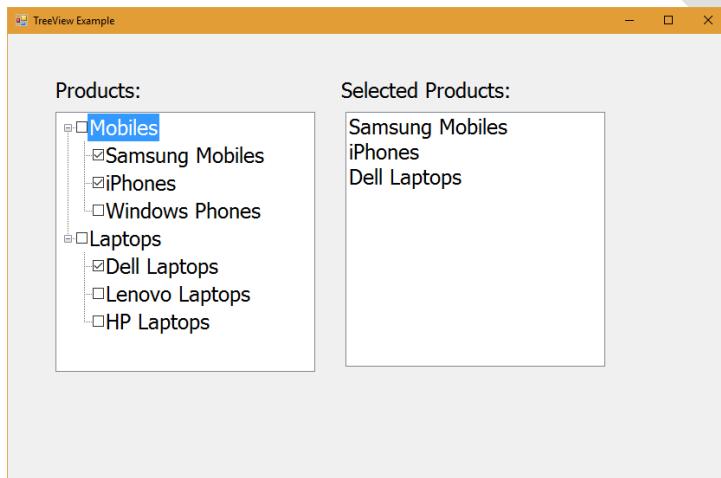
Sl. No	Property	Description
1	Text	<p>Represents text of the current node.</p> <p><u>Syntax:</u> <code>referencevariable.Nodes[index].Text = "string here";</code></p>

2	FullPath	Represents full path of the current node. <u>Syntax:</u> <code>referencevariable.Nodes[index].FullPath</code>
3	Index	Represents index of the current node. Index starts from zero (0). <u>Syntax:</u> <code>referencevariable.Nodes[index].Index</code>
4	Level	Represents level of the current node. Index starts from zero (0). <u>Syntax:</u> <code>referencevariable.Nodes[index].Level</code>
5	Nodes	Represents the collection of nodes, that should be appear in the tree view. <u>Syntax:</u> <code>referencevariable.Nodes.AddRange(node array here);</code> <u>Syntax:</u> <code>referencevariable.Nodes[index]</code>
6	Checked	Represents the current status of the checkbox of current tree node. <u>True:</u> The node is currently checked. <u>False:</u> The node is currently unchecked. <u>Syntax:</u> <code>referencevariable.Nodes[index].Checked = true false;</code>
7	NodeFont	Represents font settings of the current node. <u>Syntax:</u> <code>referencevariable.Nodes[index].NodeFont = new System.Drawing.Font(string FontName, int FontSize);</code>
8	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.Nodes[index].BackColor = System.Drawing.Color.Green;</code>
9	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.Nodes[index].ForeColor = System.Drawing.Color.Green;</code>

10	IsVisible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <i>referencevariable.Nodes[index].IsVisible = true false;</i></p>
----	-----------	---

“System.Windows.Forms. TreeView” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TreeViewExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TreeViewExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;

namespace TreeViewExample
{
    public partial class Form1 : Form
    {
        //create reference variables
        Label label1, label2;
        ListBox listBox1;
        TreeView treeview1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Text = "TreeView Example";
            this.Size = new Size(400, 270);
            this.Font = new Font("Tahoma", 18);

            /* creating label1 */
            label1 = new Label();
            label1.Text = "Products: ";
            label1.AutoSize = true;
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* creating label2 */
            label2 = new Label();
            label2.Text = "Selected Products: ";
            label2.AutoSize = true;
            label2.Location = new Point(380, 50);
            this.Controls.Add(label2);
        }
    }
}
```

```

/* creating treeview1 */
treeview1 = new TreeView();
treeview1.Size = new Size(300, 300);
treeview1.Location = new Point(55, 90);
treeview1.Nodes.Add("Mobiles");
treeview1.Nodes.Add("Laptops");
treeview1.Nodes[0].Nodes.Add("Samsung Mobiles");
treeview1.Nodes[0].Nodes.Add("iPhones");
treeview1.Nodes[0].Nodes.Add("Windows Phones");
treeview1.Nodes[1].Nodes.Add("Dell Laptops");
treeview1.Nodes[1].Nodes.Add("Lenovo Laptops");
treeview1.Nodes[1].Nodes.Add("HP Laptops");
treeview1.ExpandAll();
treeview1.CheckBoxes = true;
treeview1.AfterCheck += Treeview1_AfterCheck;
this.Controls.Add(treeview1);

/* creating listBox1 */
listbox1 = new ListBox();
listbox1.Size = new Size(300, 300);
listbox1.Location = new Point(390, 90);
this.Controls.Add(listbox1);
}

private void Treeview1_AfterCheck(object sender, TreeViewEventArgs e)
{
    listbox1.Items.Clear();
    for (int i = 0; i < treeview1.Nodes.Count; i++)
    {
        for (int j = 0; j < treeview1.Nodes[i].Nodes.Count; j++)
        {
            if (treeview1.Nodes[i].Nodes[j].Checked == true)
                listbox1.Items.Add(treeview1.Nodes[i].Nodes[j].Text);
        }
    }
}

```

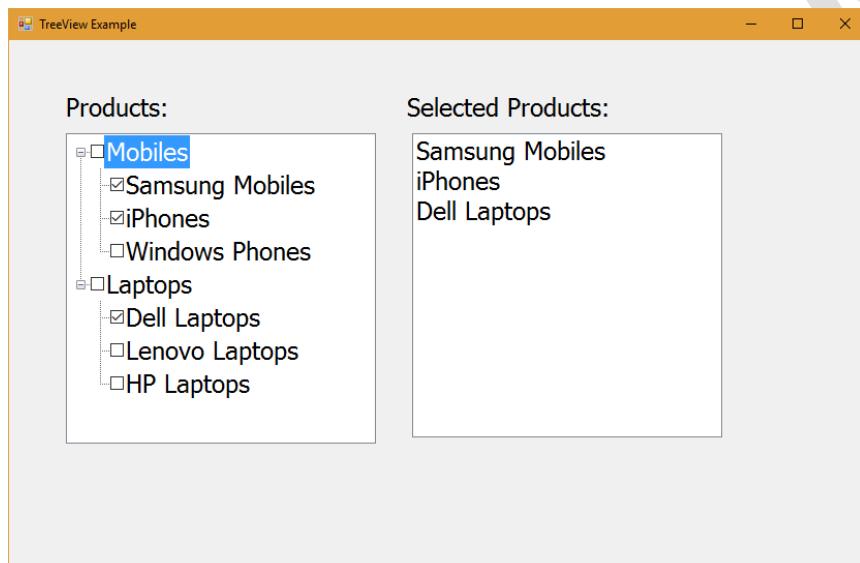
```

    }
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

When you select any node in the first treeview, only the selected names will appear in the second listbox.

The “System.Windows.Forms. PictureBox” class**The “System.Windows.Forms.PictureBox” class**

- “PictureBox” is a class; “System.Windows.Forms” is a namespace.
- The “PictureBox” class / control is used to display an image in the windows form.
- Note: You must copy and paste the image into a specific folder. Ex: C:\CSharp

Steps for development of PictureBox:

- Import the namespace:
 - `using System.Windows.Forms;`

- Create a reference variable:
 - `PictureBox referencevariable,`

- Create an object:
 - `referencevariable = new PictureBox();`

- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event += method;`

- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "PictureBox" class:

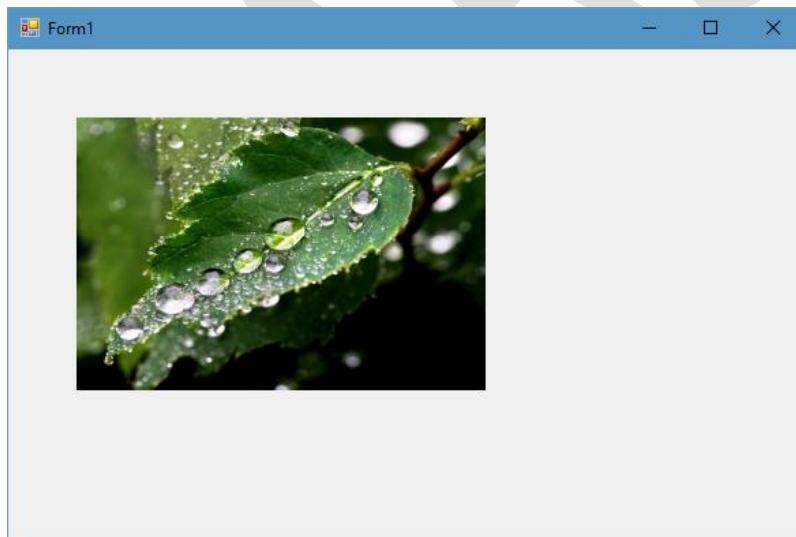
Sl. No	Property	Description
1	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code><i>referencevariable</i>.Size = new System.Drawing.Size(int width, int height);</code></p>

2	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
3	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
4	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
5	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
6	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>
7	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
8	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
9	Image	<p>Represents the image that is to be displayed in the picturebox..</p> <p><u>Syntax:</u> <code>referencevariable.Image = System.Drawing.Image.FromFile("image file path");</code></p>

10	SizeMode <u>Options:</u> Normal CenterImage StretchImage Zoom AutoSize <u>Syntax:</u> <i>referencevariable.SizeMode</i> = <i>System.Windows.Forms.PictureBoxSizeMode.optionhere;</i>
----	--

Events of “PictureBox” class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the picturebox. <u>Syntax:</u> <i>referencevariable.Click += methodname;</i>

“System.Windows.Forms. PictureBox” class - Example**Expected Output****Creating Project**

- Create a folder called “CSharp” in “C:\”. Copy and paste “img1.jpg” into “C:\CSharp” folder.
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “PictureBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “PictureBoxExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

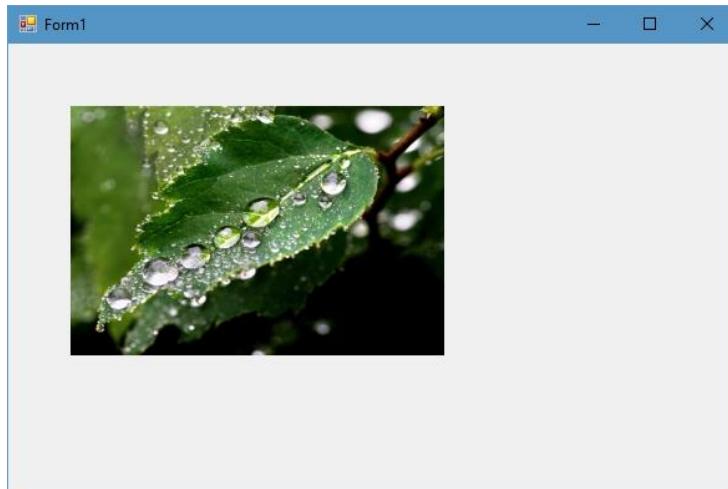
```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PictureBoxExample
{
    public partial class Form1 : Form
    {
        PictureBox pb1;
        public Form1()
        {
            InitializeComponent();
            this.Size = new Size(600, 400);
            pb1 = new PictureBox();
            pb1.Size = new Size(300, 200);
            pb1.Location = new Point(50, 50);
            pb1.Image = Image.FromFile(@"C:\CSharp\img1.jpg");
            pb1.SizeMode = PictureBoxSizeMode.StretchImage;
            this.Controls.Add(pb1);
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

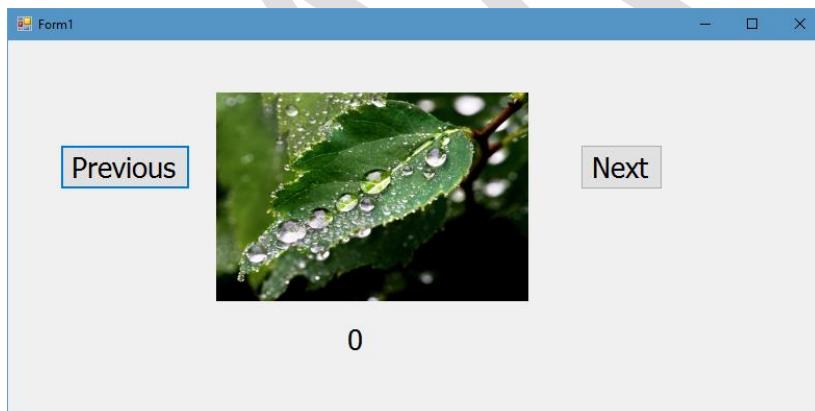
Output



The image appears in the form.

"System.Windows.Forms. PictureBox" class – Example 2

Expected Output



Creating Project

- Create a folder called "CSharp" in "C:\\". Copy and paste "img1.jpg" to "img10.jpg" into "C:\CSharp" folder.
- Open Visual Studio 2019. Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8". Select "Visual C#".

- Select “Windows Forms Application”.
- Type the project name as “PictureBoxExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “PictureBoxExample2”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PictureBoxExample2
{
    public partial class Form1 : Form
    {
        PictureBox pictureBox1;
        Button button1, button2;
        Label lbl1;
        string[] myimages = new string[] { @"C:\CSharp\img1.jpg",
 @"C:\CSharp\img2.jpg", @"C:\CSharp\img3.jpg", @"C:\CSharp\img4.jpg",
 @"C:\CSharp\img5.jpg", @"C:\CSharp\img6.jpg", @"C:\CSharp\img7.jpg",
 @"C:\CSharp\img8.jpg", @"C:\CSharp\img9.jpg", @"C:\CSharp\img10.jpg" };

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(800, 400);

            /*pictureBox*/
            pictureBox1 = new PictureBox();
            pictureBox1.Size = new Size(300, 200);
            pictureBox1.Location = new Point(200, 50);
```

```
pictureBox1.Image = Image.FromFile(myimages[0]);
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
this.Controls.Add(pictureBox1);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Previous";
button1.Location = new Point(50, 100);
button1.Click += Button1_Click;
this.Controls.Add(button1);

/* button2 */
button2 = new Button();
button2.AutoSize = true;
button2.Text = "Next";
button2.Location = new Point(550, 100);
button2.Click += Button2_Click;
this.Controls.Add(button2);

/* lbl1 */
lbl1 = new Label();
lbl1.AutoSize = true;
lbl1.Text = "0";
lbl1.Location = new Point(320, 270);
this.Controls.Add(lbl1);
}

private void Button1_Click(object sender, EventArgs e)
{
    int n = Convert.ToInt32(lbl1.Text);
    n--;
    if (n < 0)
    {
        n = 0;
        MessageBox.Show("Already at first image");
    }
    lbl1.Text = Convert.ToString(n);
```

```

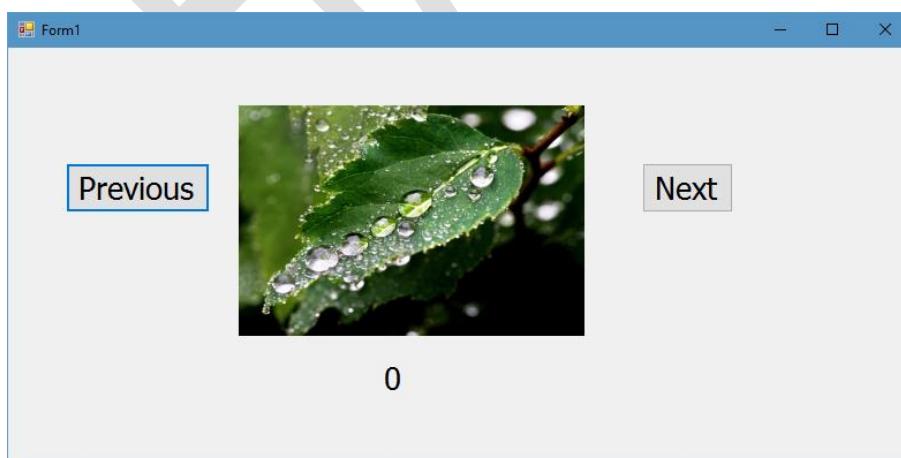
string imgpath = myimages[n];
Image img = Image.FromFile(imgpath);
pictureBox1.Image = img;
}
private void Button2_Click(object sender, EventArgs e)
{
    int n = Convert.ToInt32(lbl1.Text);
    n++;
    if (n == myimages.Length)
    {
        n = myimages.Length - 1;
        MessageBox.Show("Already at last image");
    }
    lbl1.Text = Convert.ToString(n);
    string imgpath = myimages[n];
    Image img = Image.FromFile(imgpath);
    pictureBox1.Image = img;
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Previous” button to go to previous image.

Click on “Next” button to go to next image.

The “System.Windows.Forms.Panel” class

The “System.Windows.Forms.Panel” class

- “Panel” is a class; “System.Windows.Forms” is a namespace.
- The “Panel” class / control is used to display a container in the windows form. Inside the panel, you can place any controls.
- We can divide the form as parts. Each part is a panel.

Steps for development of Panel:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `Panel referencevariable;`
- Create an object:
 - `referencevariable = new Panel();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:

- `this.Controls.Add(referencevariable);`

Properties of "Panel" class:

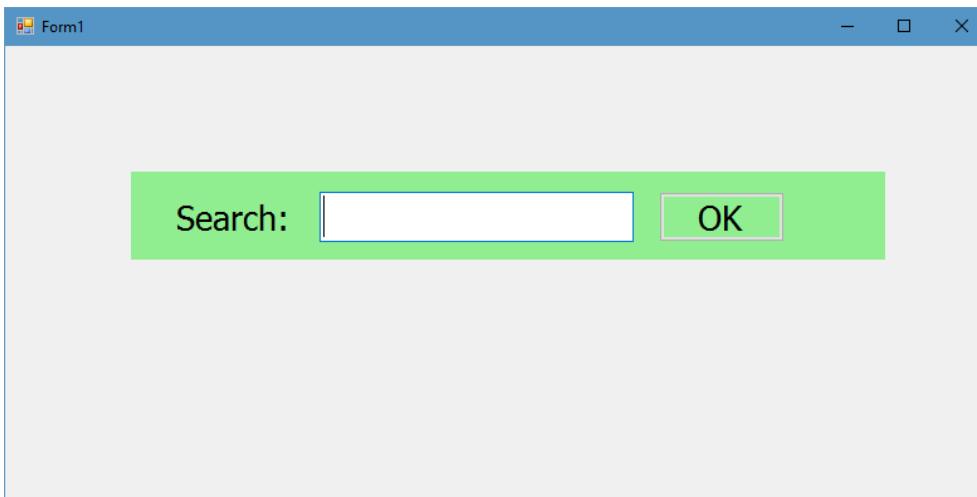
Sl. No	Property	Description
1	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
2	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
3	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
4	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
5	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
6	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

7	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
8	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
9	Controls	<p>Represents the collection of controls that should be appear inside the panel.</p> <p><u>Syntax:</u> <code>referencevariable.Controls.Add(control here);</code></p> <p><u>Syntax:</u> <code>referencevariable.Controls[index]</code></p>
10	AutoScroll	<p><u>True:</u> Displays scrollbar in the panel, if required.</p> <p><u>False:</u> Don't display scrollbar in the panel.</p> <p><u>Syntax:</u> <code>referencevariable.AutoScroll = true false;</code></p>

Events of "Panel" class:

Sl. No	Event	Description
1	Click	<p>Executes when the user clicks on the panel.</p> <p><u>Syntax:</u> <code>referencevariable.Click += methodname;</code></p>

"System.Windows.Forms.Panel" class - Example**Expected Output**



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “PanelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “PanelExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PanelExample
{
    public partial class Form1 : Form
    {
```

```
Panel panel1;
Label lbl1;
TextBox txt1;
Button btn1;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(800, 400);

    /* panel1 */
    panel1 = new Panel();
    panel1.Size = new Size(600, 70);
    panel1.Location = new Point(100, 100);
    panel1.BackColor = Color.LightGreen;
    this.Controls.Add(panel1);

    /* lbl1 */
    lbl1 = new Label();
    lbl1.AutoSize = true;
    lbl1.Text = "Search:";
    lbl1.Location = new Point(30, 20);
    panel1.Controls.Add(lbl1);

    /* txt1 */
    txt1 = new TextBox();
    txt1.Location = new Point(150, 16);
    txt1.Size = new Size(250, 40);
    panel1.Controls.Add(txt1);

    /* btn1 */
    btn1 = new Button();
    btn1.Text = "OK";
    btn1.Location = new Point(420, 16);
```

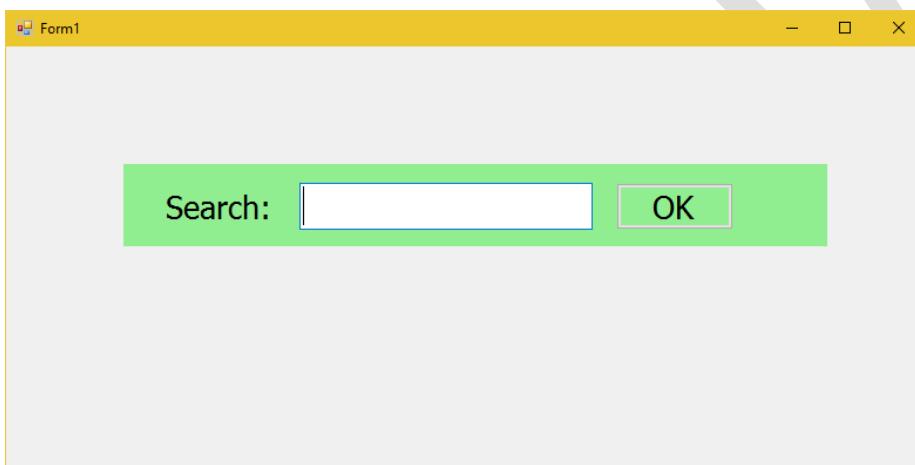
```

    btn1.Size = new Size(100, 40);
    panel1.Controls.Add(btn1);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

It shows panel in the form.

It shows label, textbox and button in the panel.

The “System.Windows.Forms.GroupBox” class**The “System.Windows.Forms.GroupBox” class**

- “GroupBox” is a class; “System.Windows.Forms” is a namespace.
- The “GroupBox” class / control is used to display a container in the windows form. Inside the groupbox, you can place any controls.
- Groupbox is a “panel”, with a title and border.

Steps for development of GroupBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `GroupBox referencevariable,`
- Create an object:
 - `referencevariable = new GroupBox();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "GroupBox" class:

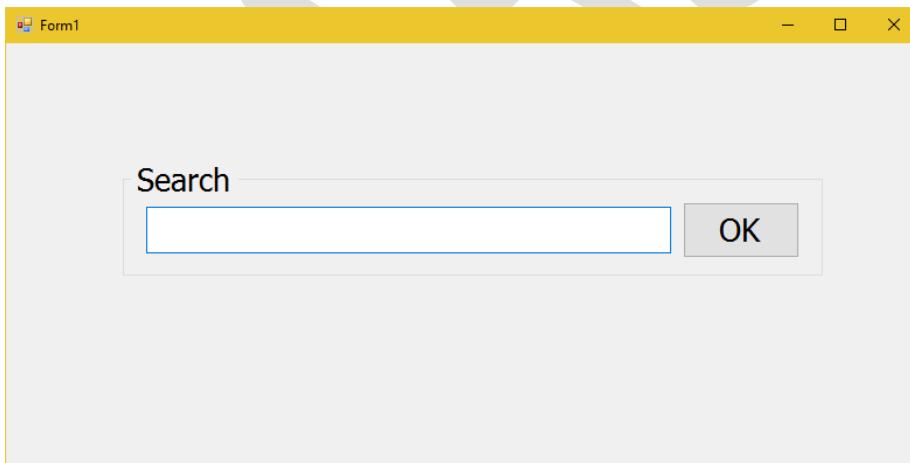
Sl. No	Property	Description
1	<code>Text</code>	Represents title of the groupbox. <u>Syntax:</u> <code><i>referencevariable.Text</i> = "any text here";</code>

2	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>pointername.Size = new System.Drawing.Size(int width, int height);</code></p>
3	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
4	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
5	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>pointername.BackColor = System.Drawing.Color.Green;</code></p>
6	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
7	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>
8	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
9	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number,</code></p>

10	Controls	Represents the collection of controls that should be appear inside the groupbox. <u>Syntax:</u> <code>referencevariable.Controls.Add(control here);</code> <u>Syntax:</u> <code>referencevariable.Controls[index]</code>
----	----------	--

Events of “GroupBox” class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the groupbox. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>

“System.Windows.Forms. GroupBox” class - Example**Expected Output****Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Windows Forms Application”.
- Type the project name as “GroupBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “GroupBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace GroupBoxExample
{
    public partial class Form1 : Form
    {
        GroupBox groupbox1;
        TextBox txt1;
        Button btn1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(800, 400);

            /* panel1 */
            groupbox1 = new GroupBox();
            groupbox1.Text = "Search";
            groupbox1.Size = new Size(600, 100);
            groupbox1.Location = new Point(100, 100);
            this.Controls.Add(groupbox1);
        }
    }
}
```

```
/* txt1 */
txt1 = new TextBox();
txt1.Location = new Point(20, 40);
txt1.Size = new Size(450, 44);
groupbox1.Controls.Add(txt1);

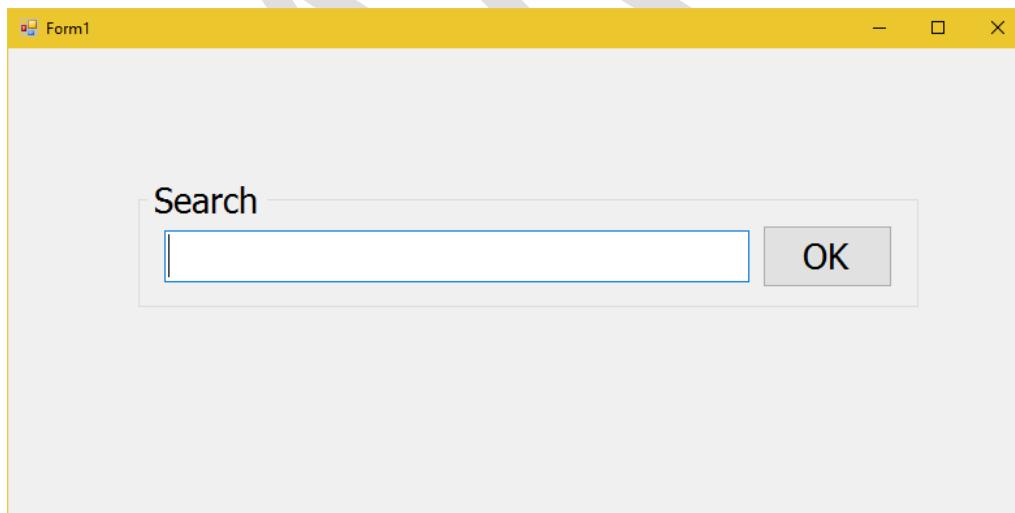
/* btn1 */
btn1 = new Button();
btn1.Text = "OK";
btn1.Location = new Point(480, 36);
btn1.Size = new Size(100, 48);
groupbox1.Controls.Add(btn1);
}

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows groupbox in the form.

It shows textbox and button in the groupbox.

The “System.Windows.Forms.SplitContainer” class

- “SplitContainer” is a class; “System.Windows.Forms” is a namespace.
- The “SplitContainer” class / control is used to divide the form as two parts and allow the user to resize the parts.
- Inside the SplitContainer two panels (panel1 and panel2) will be created automatically; you can place any controls in panel1 or panel2.

Steps for development of SplitContainer:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `SplitContainer referencevariable;`
- Create an object:
 - `referencevariable = new SplitContainer();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “SplitContainer” class:

Sl. No	Property	Description
1	Panel1	Represents first panel (left side panel) in the splitcontainer. <u>Syntax:</u> <code>referencevariable.Panel1</code>
2	Panel2	Represents second panel (right side panel) in the splitcontainer. <u>Syntax:</u> <code>referencevariable.Panel2</code>
3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
4	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code>
5	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code>
6	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
7	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>

8	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
9	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
10	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>pointername.TabIndex = any number;</i>
11	Controls	Represents the collection of controls that should be appear inside the groupbox. <u>Syntax:</u> <i>referencevariable.Controls.Add(control here);</i> <u>Syntax:</u> <i>referencevariable.Controls[index]</i>
12	Dock	Displays the splitcontainer at most top / right / bottom / left side / full screen of the form. <u>Options:</u> None Fill Top Right Bottom Left <u>Syntax:</u> <i>referencevariable.Dock = System.Windows.Forms.DockStyle.option here;</i>

Events of "SplitContainer" class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the splitcontainer. <u>Syntax:</u> <i>referencevariable.Click += methodname;</i>

“System.Windows.Forms.SplitContainer” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SplitContainerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SplitContainerExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace SplitContainerExample
```

```
{  
    public partial class Form1 : Form  
    {  
        SplitContainer splitcontainer1;  
        Label lbl1, lbl2;  
  
        public Form1()  
        {  
            InitializeComponent();  
  
            /* form properties */  
            this.Font = new Font("Tahoma", 20);  
            this.Size = new Size(800, 400);  
  
            /* panel1 */  
            splitcontainer1 = new SplitContainer();  
            splitcontainer1.Size = new Size(600, 100);  
            splitcontainer1.Location = new Point(100, 100);  
            splitcontainer1.Panel1.BackColor = Color.LightBlue;  
            splitcontainer1.Panel2.BackColor = Color.LightGreen;  
            splitcontainer1.Panel1.AutoScroll = true;  
            splitcontainer1.Panel2.AutoScroll = true;  
            splitcontainer1.Dock = DockStyle.Fill;  
            this.Controls.Add(splitcontainer1);  
  
            /* lbl1 */  
            lbl1 = new Label();  
            lbl1.AutoSize = true;  
            lbl1.Text = "Label1 in Panel1";  
            lbl1.Location = new Point(20, 20);  
            splitcontainer1.Panel1.Controls.Add(lbl1);  
  
            /* lbl2 */  
            lbl2 = new Label();  
            lbl2.AutoSize = true;  
            lbl2.Text = "Label2 in Panel1";  
            lbl2.Location = new Point(20, 20);  
        }  
    }  
}
```

```

    splitcontainer1.Panel2.Controls.Add(lbl2);
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The form is divided as two panels (parts).

We can resize the panels, by dragging in the middle of splitcontainer.

The “System.Windows.Forms.TabControl” class

The “System.Windows.Forms.TabControl” class

- “TabControl” is a class; “System.Windows.Forms” is a namespace.
- The “TabControl” class / control is used to divide the form as two tabs and displays only one tab at-a-time on the screen.
- TabControl is a collection of tabpages. An object of tabpage represents one tab.

Steps for development of TabControl:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `TabControl referencevariable,`
- Create an object:
 - `referencevariable = new TabControl();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "TabControl" class:

Sl. No	Property	Description
1	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code><i>referencevariable.Size</i> = new System.Drawing.Size(int width, int height);</code></p>

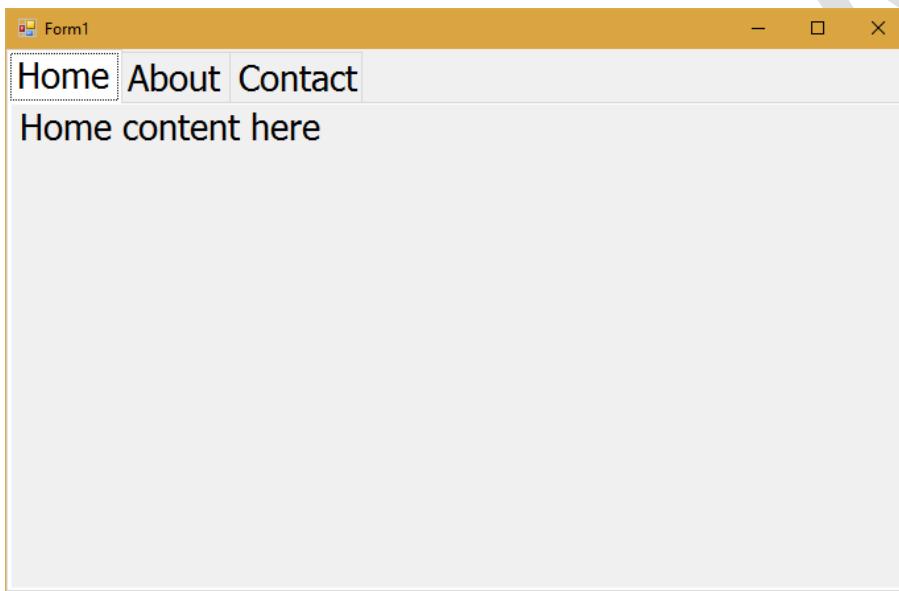
2	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
3	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
4	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>
5	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
6	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
7	Controls	<p>Represents the collection of controls that should appear inside the tabcontrol.</p> <p><u>Syntax:</u> <code>referencevariable.Controls.Add(control here);</code></p> <p><u>Syntax:</u> <code>referencevariable.Controls[index]</code></p>
8	Dock	<p>Displays the tabcontrol at most top / right / bottom / left side / full screen of the form.</p> <p><u>Options:</u> None Fill Top Right Bottom Left</p> <p><u>Syntax:</u> <code>referencevariable.Dock = System.Windows.Forms.DockStyle.option here;</code></p>

Events of "TabControl" class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the tabcontrol. <u>Syntax:</u> <i>referencevariable.Click += methodname;</i>

“System.Windows.Forms.TabControl” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TabControlExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TabControlExample”.
- Click on OK.

- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace TabControlExample
{
    public partial class Form1 : Form
    {
        TabControl tabcontrol1;
       TabPage tabPage1, tabPage2, tabPage3;
        Label lbl1, lbl2, lbl3;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(700, 450);

            /* tabcontrol1 */
            tabcontrol1 = new TabControl();
            tabcontrol1.Dock = DockStyle.Fill;
            this.Controls.Add(tabcontrol1);

            /* tabPage1 */
            tabPage1 = newTabPage();
            tabPage1.Text = "Home";
            tabcontrol1.Controls.Add(tabpage1);

            /* tabPage2 */
            tabPage2 = newTabPage();
```

```
tabpage2.Text = "About";
tabcontrol1.Controls.Add(tabpage2);

/* tabpage3 */
tabpage3 = new TabPage();
tabpage3.Text = "Contact";
tabcontrol1.Controls.Add(tabpage3);

/* lbl1 */
lbl1 = new Label();
lbl1.Text = "Home content here";
lbl1.AutoSize = true;
tabpage1.Controls.Add(lbl1);

/* lbl2 */
lbl2 = new Label();
lbl2.Text = "About content here";
lbl2.AutoSize = true;
tabpage2.Controls.Add(lbl2);

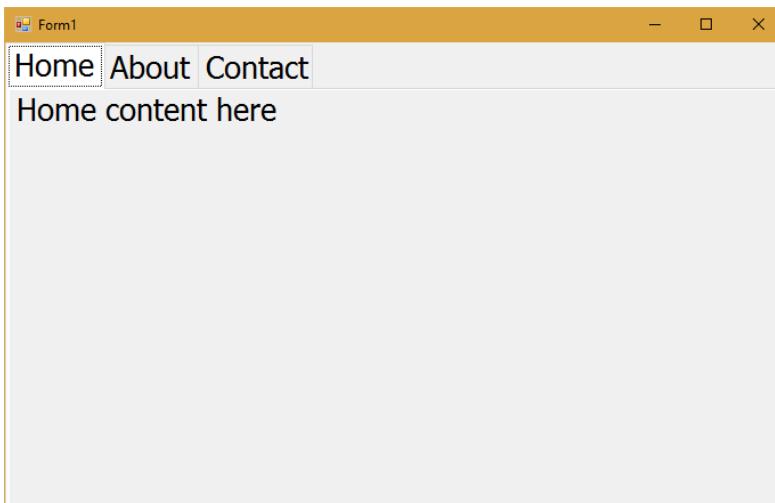
/* lbl3 */
lbl3 = new Label();
lbl3.Text = "Contact content here";
lbl3.AutoSize = true;
tabpage3.Controls.Add(lbl3);

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The form is divided as 3 tabs.

If you click on "Home", it shows "Home content here".

If you click on "About", it shows "About content here".

If you click on "Contact", it shows "Contact content here".

The "System.Windows.Forms.FlowLayoutPanel" class

- "FlowLayoutPanel" is a class; "System.Windows.Forms" is a namespace.
- The "FlowLayoutPanel" class / control is same as panel, but it arranges the controls side-by-side automatically. When a row is over, it automatically moves the control to the next line.

Steps for development of FlowLayoutPanel:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `FlowLayoutPanel referencevariable,`

- Create an object:
 - `referencevariable = new FlowLayoutPanel();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "FlowLayoutPanel" class:

Sl. No	Property	Description
1	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
2	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
3	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>

4	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
5	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
6	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>
7	Controls	Represents the collection of controls that should be appear inside the tabcontrol. <u>Syntax:</u> <i>referencevariable.Controls.Add(control here);</i> <u>Syntax:</u> <i>referencevariable.Controls[index]</i>
8	Dock	Displays the tabcontrol at most top / right / bottom / left side / full screen of the form. <u>Options:</u> None Fill Top Right Bottom Left <u>Syntax:</u> <i>referencevariable.Dock = System.Windows.Forms.DockStyle.option here;</i>
9	FlowDirection	Represents the direction of controls in the flowlayoutpanel. <u>Options:</u> LeftToRight RightToLeft TopDown BottomUp <u>Syntax:</u> <i>referencevariable.FlowDirection = System.Windows.Forms.FlowDirection.option here;</i>

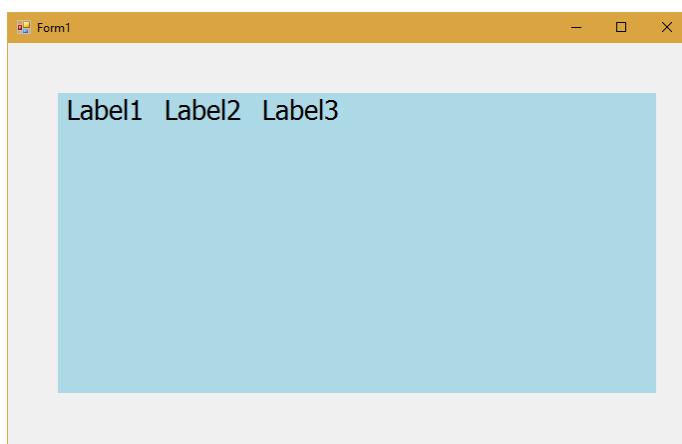
Events of "FlowLayoutPanel" class:

Sl. No	Event	Description

1	Click	<p>Executes when the user clicks on the flowlayoutpanel.</p> <p><u>Syntax:</u> <code>referencevariable.Click += methodname;</code></p>
---	-------	--

“System.Windows.Forms. FlowLayoutPanel” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FlowLayoutPanelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FlowLayoutPanelExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FlowLayoutPanelExample
{
    public partial class Form1 : Form
    {
        FlowLayoutPanel flowLayoutPanel1;
        Label lbl1, lbl2, lbl3;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(700, 450);

            /* flowLayoutPanel1 */
            flowLayoutPanel1 = new FlowLayoutPanel();
            flowLayoutPanel1.BackColor = Color.LightBlue;
            flowLayoutPanel1.Size = new Size(600, 300);
            flowLayoutPanel1.Location = new Point(50, 50);
            flowLayoutPanel1.FlowDirection = FlowDirection.LeftToRight;
            this.Controls.Add(flowLayoutPanel1);

            /* lbl1 */
            lbl1 = new Label();
            lbl1.Text = "Label1";
            lbl1.AutoSize = true;
            flowLayoutPanel1.Controls.Add(lbl1);

            /* lbl2 */
            lbl2 = new Label();
            lbl2.Text = "Label2";
            lbl2.AutoSize = true;
```

```

flowLayoutPanel1.Controls.Add(lbl2);

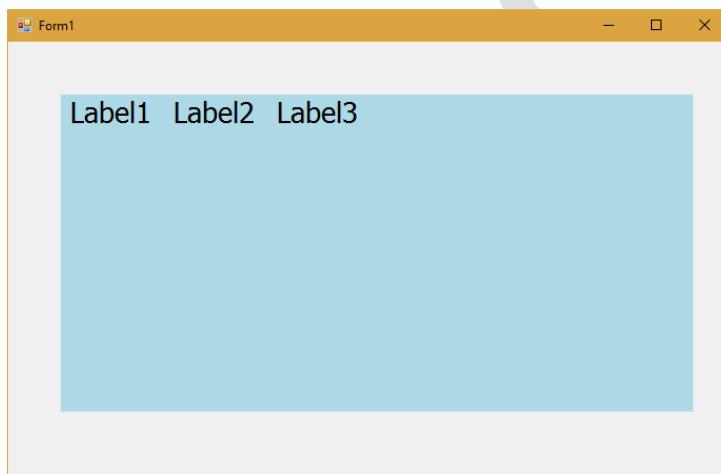
/* lbl3 */
lbl3 = new Label();
lbl3.Text = "Label3";
lbl3.AutoSize = true;
flowLayoutPanel1.Controls.Add(lbl3);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The flowLayoutPanel appears in the form.

There are 3 controls called label1, label2, label3 in the flowLayoutPanel.

“System.Windows.Forms.LinkLabel” class

The “System.Windows.Forms.LinkLabel” class

- “LinkLabel” is a class; “System.Windows.Forms” is a namespace.

- The “LinkLabel” class / control is used to display a clickable hyperlink.
- LinkLabel executes “LinkClicked” event, when the user clicks it.

Steps for development of LinkLabel:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `LinkLabel referencevariable,`
- Create an object:
 - `referencevariable = new LinkLabel();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “LinkLabel” class:

Sl. No	Property	Description

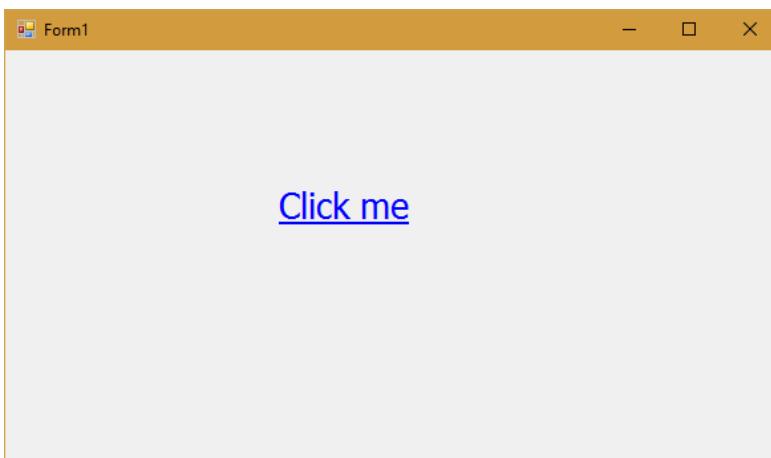
1	Text	<p>Represents text of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Text = "text here";</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	AutoSize	<p><u>True:</u> The control will take the essential size automatically.</p> <p><u>False:</u> The control will not take the size automatically; we need to set the size manually.</p> <p><u>Syntax:</u> <code>referencevariable.AutoSize = true false;</code></p>
5	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
6	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
7	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>

8	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.ForeColor = System.Drawing.Color.Green;</i>
9	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
10	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>

Events of "LinkLabel" class:

Sl. No	Event	Description
1	LinkClicked	Executes when the user clicks on the control. <u>Syntax:</u> <i>referencevariable.LinkClicked += methodname;</i>
2	MouseEnter	Executes when the user moves the mouse pointer from outside to inside the control. <u>Syntax:</u> <i>referencevariable.MouseEnter += methodname;</i>
3	MouseLeave	Executes when the user moves the mouse pointer from inside to outside the control. <u>Syntax:</u> <i>referencevariable.MouseLeave += methodname;</i>

"System.Windows.Forms. LinkLabel" class - Example**Expected Output**



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “LinkLabelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LinkLabelExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace LinkLabelExample
{
    public partial class Form1 : Form
    {
        LinkLabel linklabel1;
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```

/* form properties */
this.Font = new Font("Tahoma", 20);
this.Size = new Size(600, 350);

/* linklabel1 */
linklabel1 = new LinkLabel();
linklabel1.Text = "Click me";
linklabel1.AutoSize = true;
linklabel1.Location = new Point(200, 100);
linklabel1.LinkClicked += Linklabel1_LinkClicked;
this.Controls.Add(linklabel1);
}

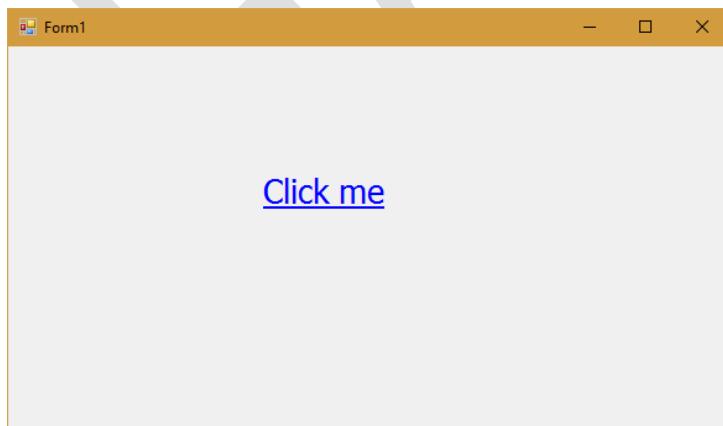
private void Linklabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    linklabel1.Text = "Clicked";
}
}
}

```

Running the Project

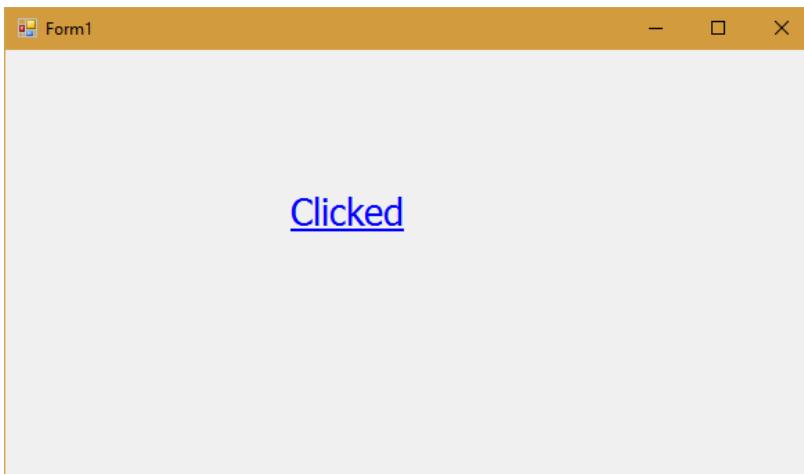
- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows “Click me” linklabel.

When you click on the linklabel, it will be changed as “Clicked”.



"System.Windows.Forms. WebBrowser" class

- "WebBrowser" is a class; "System.Windows.Forms" is a namespace.
- The "WebBrowser" class / control is used to display a web page in the form.
- It uses "Internet Explorer" internally. It requires internet connection.

Steps for development of WebBrowser:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `WebBrowser referencevariable;`
- Create an object:
 - `referencevariable = new WebBrowser();`
- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event += method;`

- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "WebBrowser" class:

Sl. No	Property	Description
1	Url	<p>Represents url (website address) of the webbrowser.</p> <p><u>Syntax:</u> <code>referencevariable.url = new Uri("url here");</code></p>
2	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
3	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
4	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

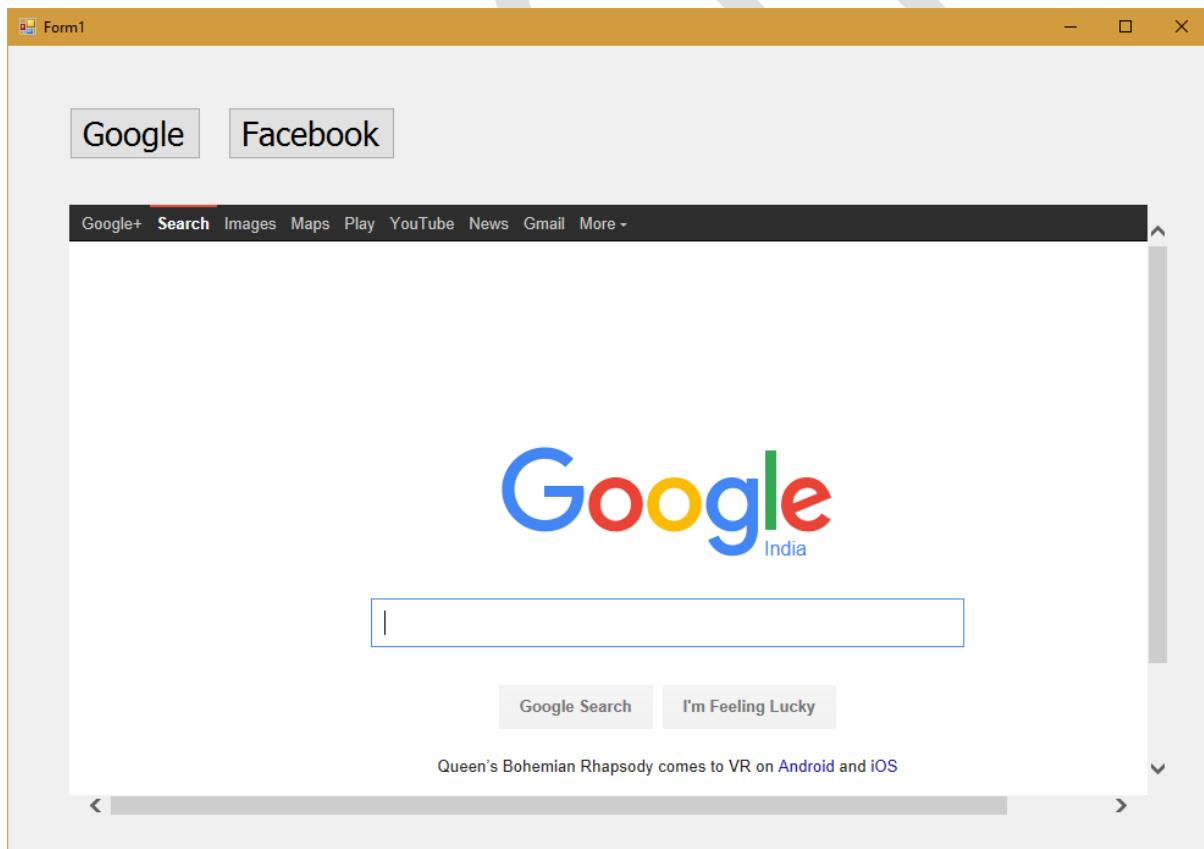
Methods of "WebBrowser" class:

Sl. No	Method	Description

1	Navigate()	It navigates to the specified url. <u>Syntax:</u> <i>referencevariable.Navigate("url")</i>
2	GoBack()	Navigates to the previous page. <u>Syntax:</u> <i>referencevariable.GoBack()</i>
3	GoForward()	Navigates to the forward page. <u>Syntax:</u> <i>referencevariable.GoForward()</i>

“System.Windows.Forms.WebBrowser” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.

- Select “Windows Forms Application”.
- Type the project name as “WebBrowserExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WebBrowser”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WebBrowserExample
{
    public partial class Form1 : Form
    {
        Button btn1, btn2;
        WebBrowser webbrowser1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(1000, 700);

            /* btn1 */
            btn1 = new Button();
            btn1.Text = "Google";
            btn1.AutoSize = true;
            btn1.Location = new Point(50, 50);
            btn1.Click += Btn1_Click;
            this.Controls.Add(btn1);

            /* btn2 */
        }
    }
}
```

```
btn2 = new Button();
btn2.Text = "Facebook";
btn2.AutoSize = true;
btn2.Location = new Point(180, 50);
btn2.Click += Btn2_Click;
this.Controls.Add(btn2);

/* webbrowser1 */
webbrowser1 = new WebBrowser();
webbrowser1.Navigate("http://www.google.com");
webbrowser1.Size = new Size(900, 500);
webbrowser1.Location = new Point(50, 130);
this.Controls.Add(webbrowser1);

}

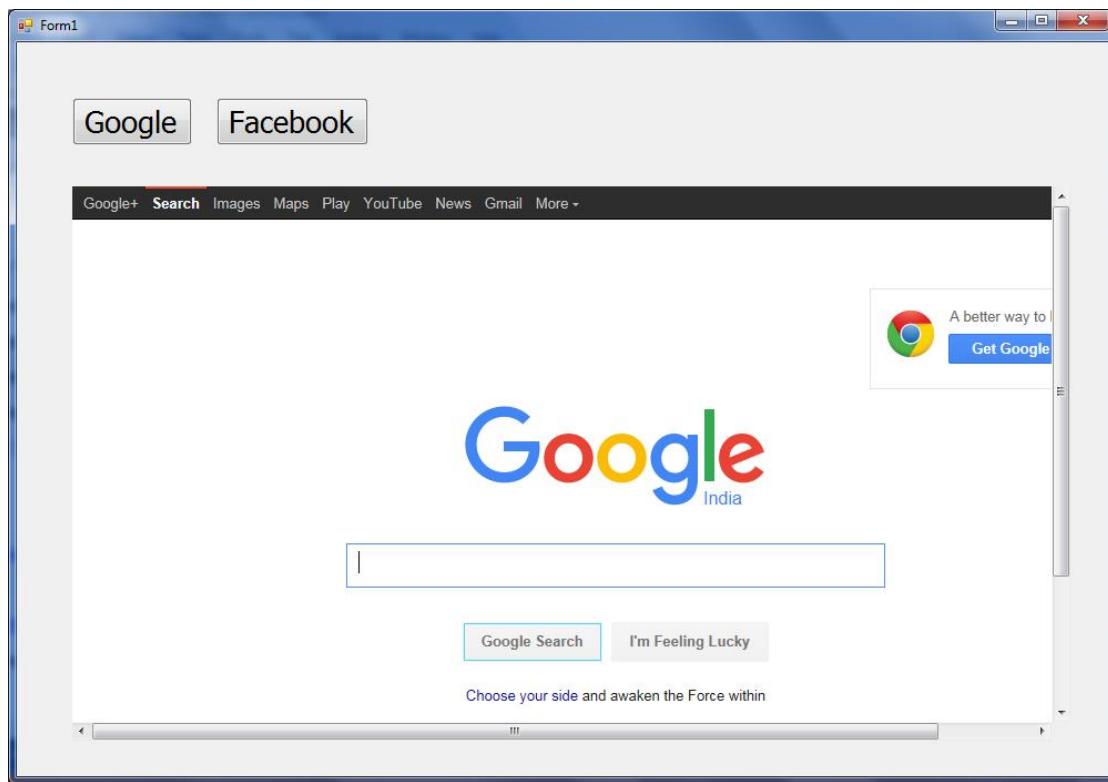
private void Btn1_Click(object sender, EventArgs e)
{
    string s = "http://www.google.com";
    webbrowser1.Navigate(s);
}

private void Btn2_Click(object sender, EventArgs e)
{
    string s = "http://www.facebook.com";
    webbrowser1.Navigate(s);
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



If you click on “Google”, it shows google web site.

If you click on “Facebook”, it shows facebook web site.

“System.Windows.Forms.Timer” class

The “System.Windows.Forms.Timer” class

- “Timer” is a class; “System.Windows.Forms” is a namespace.
- The “Timer” class / control is used to execute the code repeatedly for every ‘n’ no. of milli seconds..
- 1000 milli seconds = 1 second

Steps for development of Timer:

- Import the namespace:

- `using System.Windows.Forms;`

- Create a reference variable:
 - `Timer referencevariable;`

- Create an object:
 - `referencevariable = new Timer();`

- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event += method;`

Properties of "Timer" class:

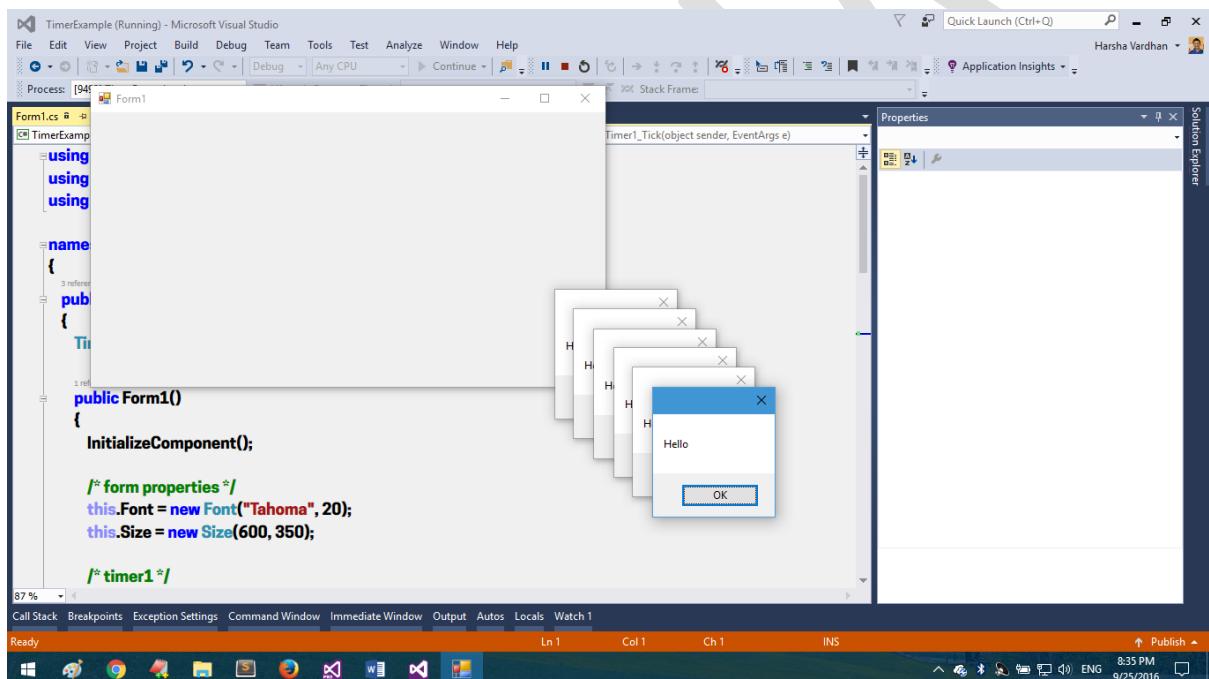
Sl. No	Property	Description
1	Interval	<p>Represents no. of milli seconds, based on which the timer's code should execute.</p> <p><u>Syntax:</u> <code>referencevariable.Interval = any number;</code></p>
2	Enabled	<p><u>True:</u> Timer is activated. It calls Tick event at the specified milli seconds.</p> <p><u>False:</u> Timer is deactivated. It doesn't call Tick event at the specified milli seconds.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>

Events of "Timer" class:

Sl. No	Event	Description
1	Tick	Executes for every completion of specified milli seconds. <u>Syntax:</u> <i>referencevariable.Tick += methodname;</i>

“System.Windows.Forms.Timer” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TimerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TimerExample”.

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace TimerExample
{
    public partial class Form1 : Form
    {
        Timer timer1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(600, 350);

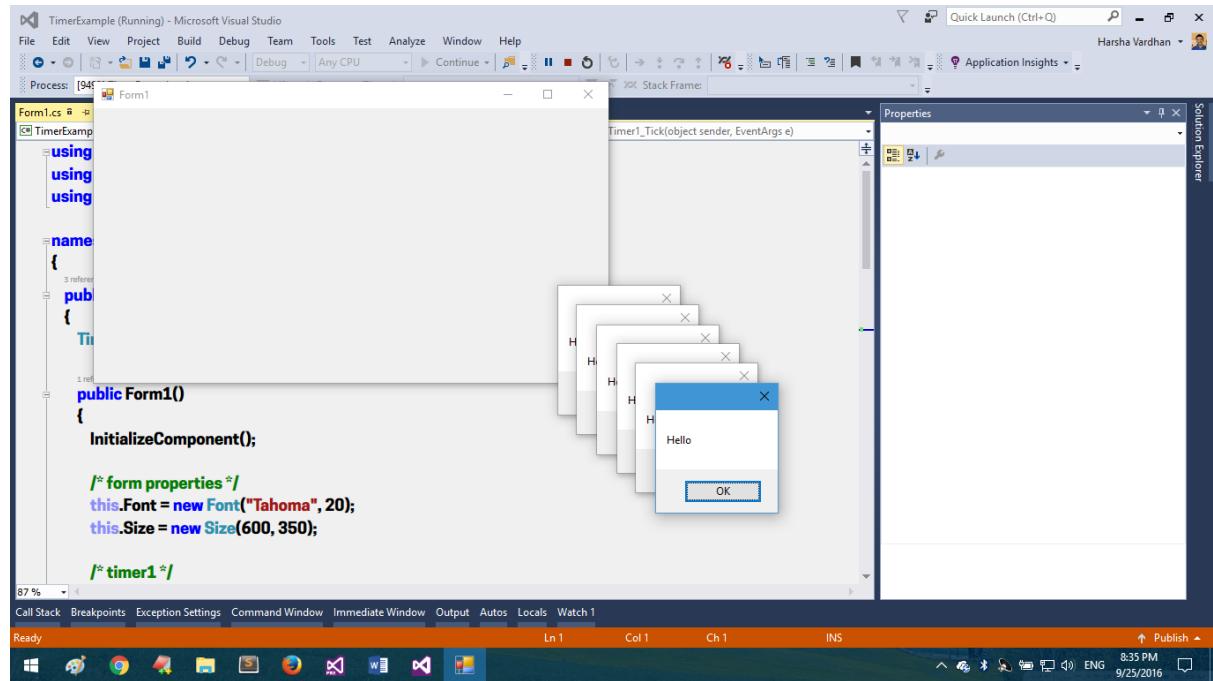
            /* timer1 */
            timer1 = new Timer();
            timer1.Enabled = true;
            timer1.Interval = 3000;
            timer1.Tick += Timer1_Tick;
        }

        private void Timer1_Tick(object sender, EventArgs e)
        {
            MessageBox.Show("Hello");
        }
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

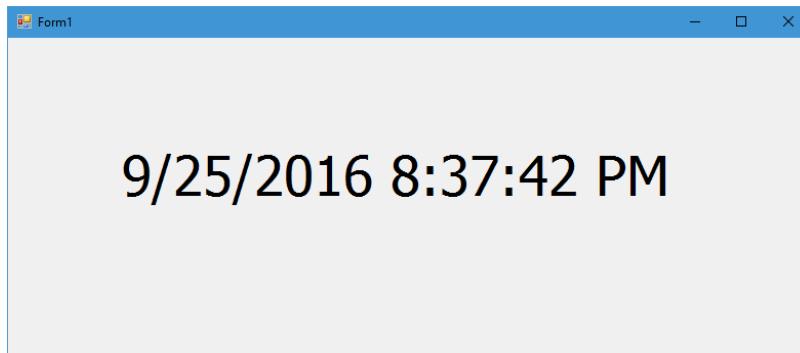
Output



For every 3 seconds, it shows “Hello” message box.

“System.Windows.Forms.Timer” class – with Time - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TimerWithTimeExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TimerWithTimeExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

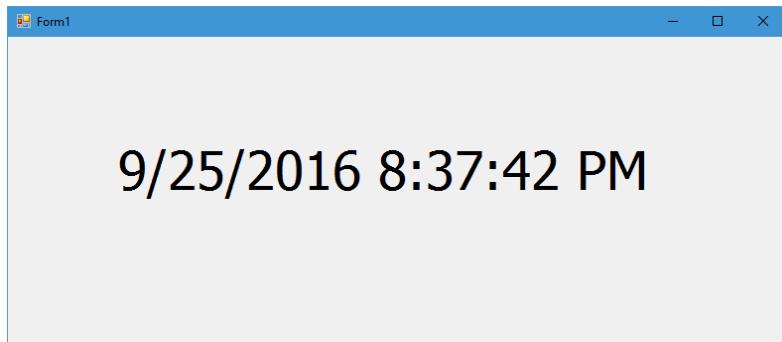
namespace TimerWithTimeExample
{
    public partial class Form1 : Form
```

```
{  
    Timer timer1;  
    Label lbl1;  
  
    public Form1()  
    {  
        InitializeComponent();  
  
        /* form properties */  
        this.Font = new Font("Tahoma", 20);  
        this.Size = new Size(800, 350);  
  
        /* lbl1 */  
        lbl1 = new Label();  
        lbl1.AutoSize = true;  
        lbl1.Text = "Time appears here";  
        lbl1.Location = new Point(100, 100);  
        lbl1.Font = new Font("Tahoma", 40);  
        this.Controls.Add(lbl1);  
  
        /* timer1 */  
        timer1 = new Timer();  
        timer1.Enabled = true;  
        timer1.Interval = 1000;  
        timer1.Tick += Timer1_Tick;  
    }  
  
    private void Timer1_Tick(object sender, EventArgs e)  
    {  
        lbl1.Text = DateTime.Now.ToString();  
    }  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

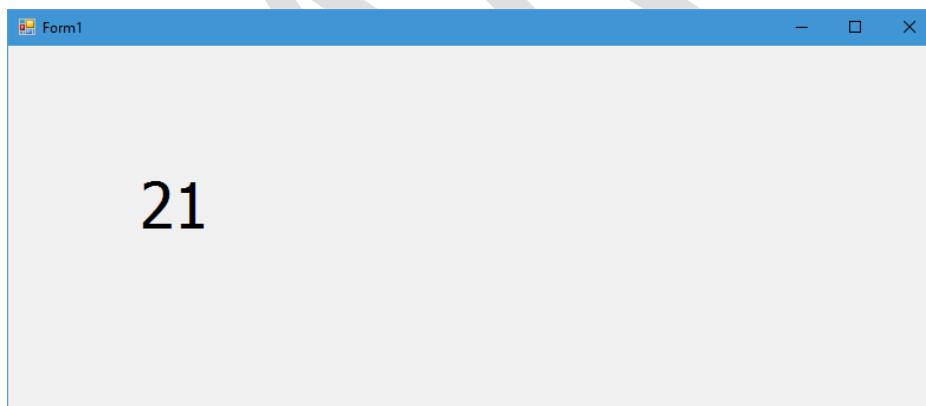
Output



It shows the system date and time in the label. It automatically updates for every second-by-second.

"System.Windows.Forms.Timer" class – with Counter - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".
- Select ".NET Framework 4.8".
- Select "Visual C#".

- Select “Windows Forms Application”.
- Type the project name as “TimerWithCounterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TimerWithCounterExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace TimerWithCounterExample
{
    public partial class Form1 : Form
    {
        Timer timer1;
        Label lbl1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(800, 350);

            /* lbl1 */
            lbl1 = new Label();
            lbl1.AutoSize = true;
            lbl1.Text = "1";
            lbl1.Location = new Point(100, 100);
            lbl1.Font = new Font("Tahoma", 40);
            this.Controls.Add(lbl1);
        }
    }
}
```

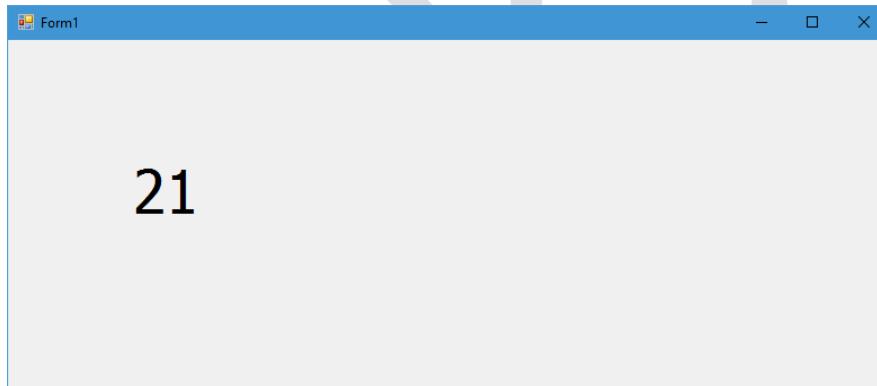
```
/* timer1 */
timer1 = new Timer();
timer1.Enabled = true;
timer1.Interval = 100;
timer1.Tick += Timer1_Tick;
}

private void Timer1_Tick(object sender, EventArgs e)
{
    lbl1.Text = Convert.ToString(Convert.ToInt32(lbl1.Text) + 1);
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

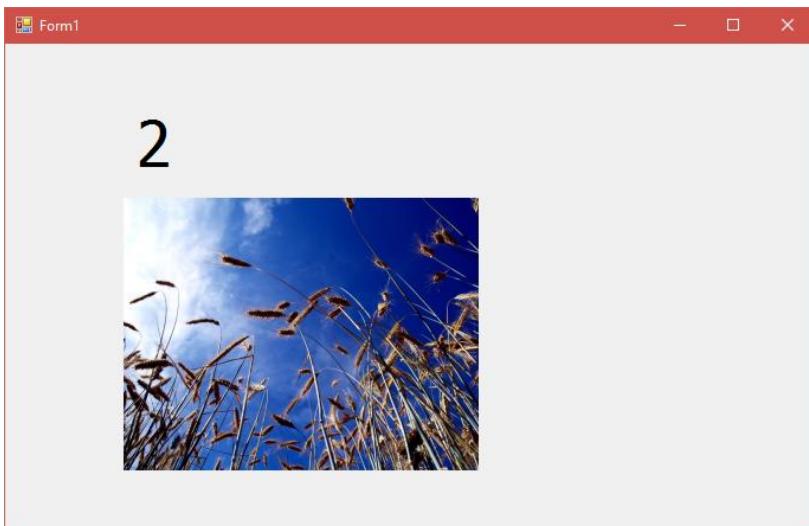
Output



It automatically increments the number in the label, for every 100 milli seconds.

“System.Windows.Forms.Timer” class – with Slide Show - Example

Expected Output



Creating Project

- Copy and paste “img1.jpg”, “img2.jpg”, ... “img10.jpg” into “C:\CSharp” folder.
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TimerSlideShowExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TimerSlideShowExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace TimerSlideShowExample
{
```

```

public partial class Form1 : Form
{
    Timer timer1;
    Label lbl1;
    PictureBox pictureBox1;

    List<string> MyImages = new List<string>() { @"C:\CSharp\img1.jpg",
    @"C:\CSharp\img2.jpg", @"C:\CSharp\img3.jpg", @"C:\CSharp\img4.jpg",
    @"C:\CSharp\img5.jpg", @"C:\CSharp\img6.jpg", @"C:\CSharp\img7.jpg",
    @"C:\CSharp\img8.jpg", @"C:\CSharp\img9.jpg", @"C:\CSharp\img10.jpg" };

    public Form1()
    {
        InitializeComponent();

        /* form properties */
        this.Font = new Font("Tahoma", 20);
        this.Size = new Size(700, 450);

        /* lbl1 */
        lbl1 = new Label();
        lbl1.AutoSize = true;
        lbl1.Text = "0";
        lbl1.Location = new Point(100, 50);
        lbl1.Font = new Font("Tahoma", 40);
        this.Controls.Add(lbl1);

        /* timer1 */
        timer1 = new Timer();
        timer1.Enabled = true;
        timer1.Interval = 1000;
        timer1.Tick += Timer1_Tick;

        /* pictureBox1 */
        pictureBox1 = new PictureBox();
        pictureBox1.Size = new Size(300, 230);
        pictureBox1.Location = new Point(100, 130);
    }
}

```

```
picturebox1.SizeMode = PictureBoxSizeMode.StretchImage;
picturebox1.Image = Image.FromFile(MyImages[0]);
this.Controls.Add(pictureBox1);
}

private void Timer1_Tick(object sender, EventArgs e)
{
    int n = Convert.ToInt32(lbl1.Text);
    n++;

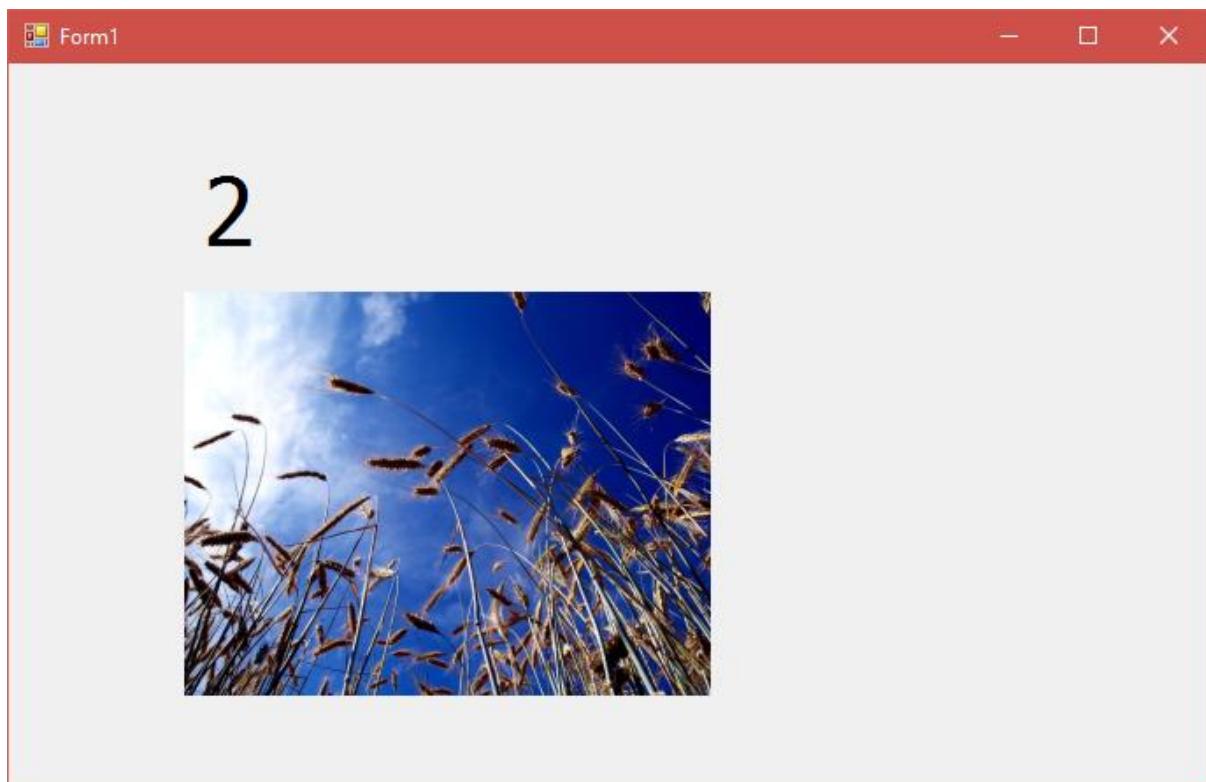
    if (n == MyImages.Count)
    {
        timer1.Enabled = false;
        MessageBox.Show("Done");
        return;
    }

    Image img = Image.FromFile(MyImages[n]);
    pictureBox1.Image = img;
    lbl1.Text = Convert.ToString(n);
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows image slide show, for every 3 seconds.

"System.Windows.Forms.ProgressBar" class

The "System.Windows.Forms.ProgressBar" class

- "ProgressBar" is a class; "System.Windows.Forms" is a namespace.
- The "ProgressBar" class / control is used to display a progress of a task.
- Look and feel depends on the operating system. Ex: Windows 7, Windows 8.1, Windows 10 etc.

Steps for development of ProgressBar:

- Import the namespace:
 - `using System.Windows.Forms;`

- Create a reference variable:
 - `ProgressBar referencevariable;`
- Create an object:
 - `referencevariable = new ProgressBar();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

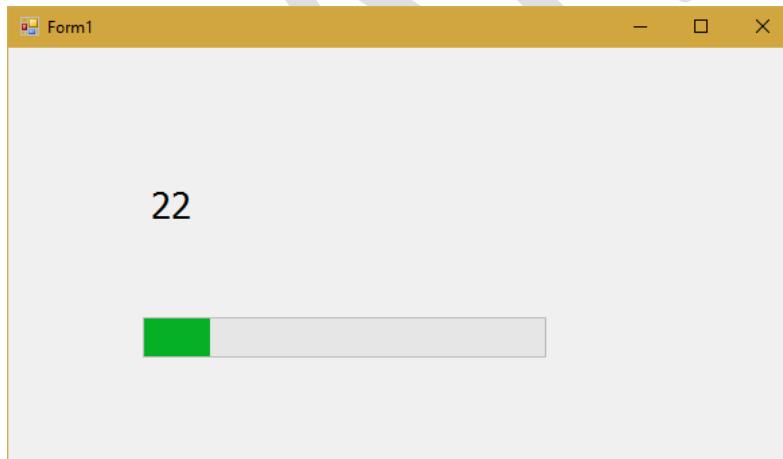
Properties of "ProgressBar" class:

Sl. No	Property	Description
1	Value	<p>Represents current value of the progress bar.</p> <p><u>Syntax:</u> <code>referencevariable.Value = any number;</code></p>
2	Minimum	<p>Represents minimum value of the progress bar.</p> <p><u>Syntax:</u> <code>referencevariable.Minimum = any number;</code></p>
3	Maximum	<p>Represents maximum value of the progress bar.</p> <p><u>Syntax:</u> <code>referencevariable.Maximum = any number;</code></p>

4	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
5	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
6	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code>
7	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code>
8	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>

Events of “ProgressBar” class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the control. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>
2	MouseEnter	Executes when the user moves the mouse pointer from outside to inside the control. <u>Syntax:</u> <code>referencevariable.MouseEnter += methodname;</code>
3	MouseLeave	Executes when the user moves the mouse pointer from inside to outside the control. <u>Syntax:</u> <code>referencevariable.MouseLeave += methodname;</code>

“System.Windows.Forms. ProgressBar” class - Example**Expected Output****Creating Project**

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ProgressBarExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ProgressBarExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace ProgressBarExample
{
    public partial class Form1 : Form
    {
        ProgressBar progressbar1;
        Timer timer1;
        Label lbl1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(600, 350);

            /* lbl1 */
            lbl1 = new Label();
            lbl1.AutoSize = true;
            lbl1.Text = "0";
        }
    }
}
```

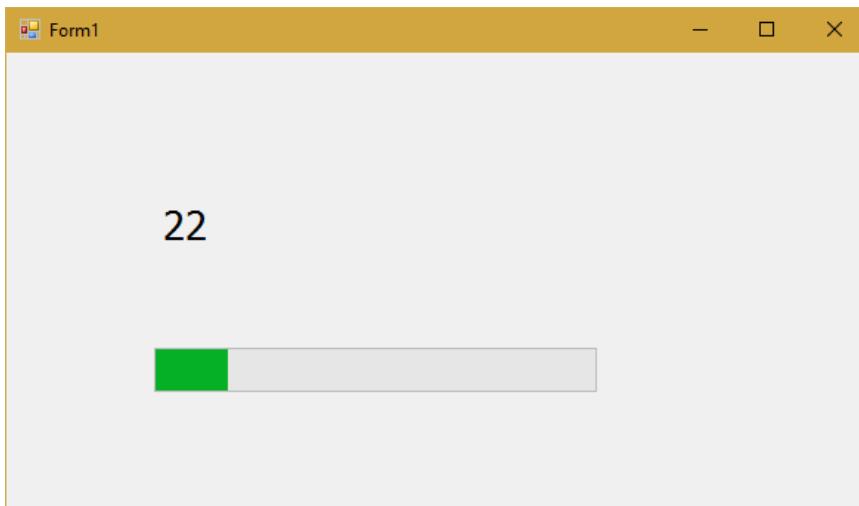
```
lbl1.Location = new Point(100, 100);
this.Controls.Add(lbl1);
/* progressbar1 */
progressbar1 = new ProgressBar();
progressbar1.Minimum = 0;
progressbar1.Maximum = 100;
progressbar1.Size = new Size(300, 30);
progressbar1.Location = new Point(100, 200);
this.Controls.Add(progressbar1);
/* timer1 */
timer1 = new Timer();
timer1.Enabled = true;
timer1.Interval = 100;
timer1.Tick += Timer1_Tick;
}
private void Timer1_Tick(object sender, EventArgs e)
{
    progressbar1.Value++;
    lbl1.Text = Convert.ToString(progressbar1.Value);

    if (progressbar1.Value >= progressbar1.Maximum)
    {
        timer1.Enabled = false;
        MessageBox.Show("Done");
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It increments progress bar automatically.

When the progress bar reaches to 100%, it shows “Done”.

“System.Windows.Forms.NotifyIcon” class

The “System.Windows.Forms.NotifyIcon” class

- “NotifyIcon” is a class; “System.Windows.Forms” is a namespace.
- The “NotifyIcon” class / control is used to display an icon in the windows taskbar, while the form is running.
- This requires a “.ico” file.

Steps for development of NotifyIcon:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `NotifyIcon referencevariable;`

- Create an object:
 - `referencevariable = new NotifyIcon();`

- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`

Properties of "NotifyIcon" class:

Sl. No	Property	Description
1	Icon	Represents the icon that should appear in the windows taskbar. <u>Syntax:</u> <code>referencevariable.Icon = new System.Drawing.Icon("icon file path");</code>
2	Visible	<u>True:</u> The control appears on the windows taskbar. <u>False:</u> The control will disappear on the windows taskbar. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>

Methods of "NotifyIcon" class:

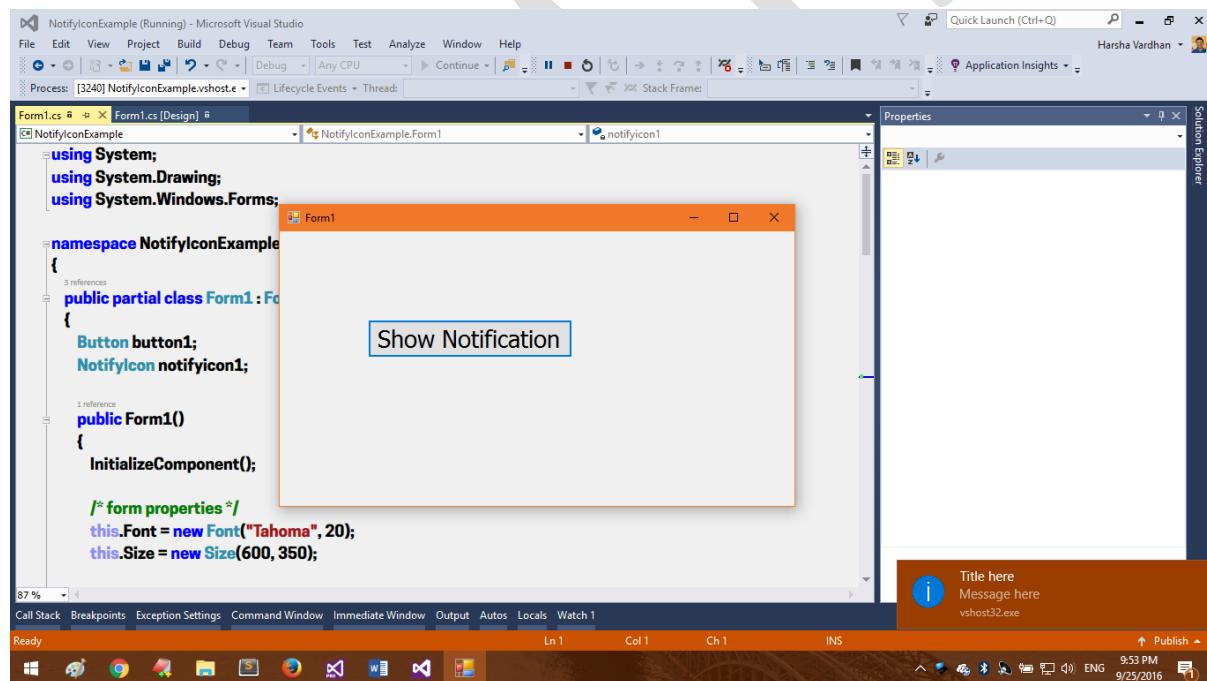
Sl. No	Method	Description
1	ShowBalloonTip	Shows a balloon tip message from the icon at windows taskbar. <u>Syntax:</u> <code>referencevariable.ShowBalloonTip(int milliseconds, string title, string message, System.Windows.Forms.ToolTipInfo Warning Error);</code>

Events of "NotifyIcon" class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the icon at windows taskbar. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>
2	BalloonTipClicked	Executes when the user clicks on the balloon tip message at windows taskbar. <u>Syntax:</u> <code>referencevariable.BalloonTipClicked += methodname;</code>

“System.Windows.Forms. NotifyIcon” class - Example

Expected Output



Creating Project

- Copy and paste “Sample.ico” file into “C:\CSharp” folder.
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “NotifyIconExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “NotifyIconExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace NotifyIconExample
{
    public partial class Form1 : Form
    {
        Button button1;
        NotifyIcon notifyicon1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(600, 350);

            /* button1 */
            button1 = new Button();
            button1.AutoSize = true;
            button1.Text = "Show Notification";
            button1.Location = new Point(100, 100);
            button1.Click += Button1_Click;
        }
    }
}
```

```
this.Controls.Add(button1);
```

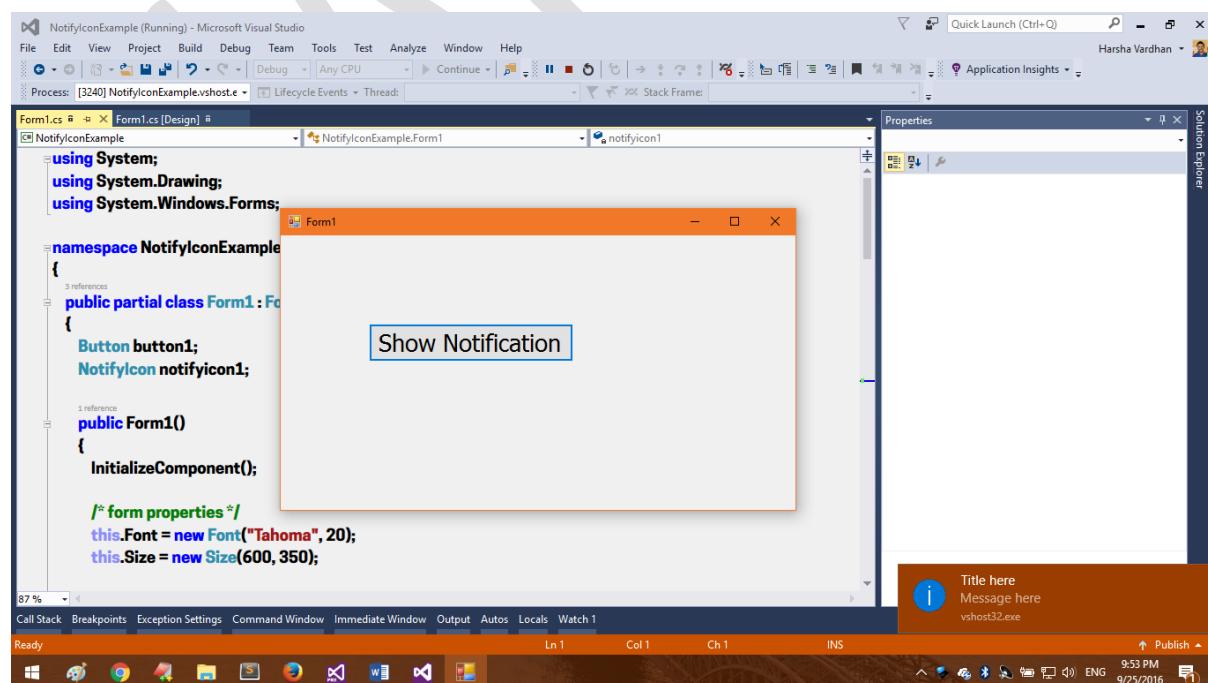
```
/* notifyicon1 */
notifyicon1 = new NotifyIcon();
notifyicon1.Icon = new Icon(@"C:\CSharp\Sample.ico");
notifyicon1.Visible = true;
}

private void Button1_Click(object sender, EventArgs e)
{
    notifyicon1.ShowBalloonTip(5000, "Title here", "Message here",
    ToolTipIcon.Info);
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows icon in the windows taskbar.

When the user clicks on the button, it shows tooltip message from the icon.

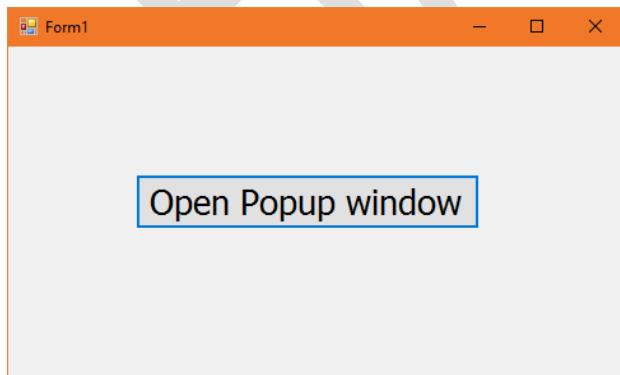
Popup Boxes

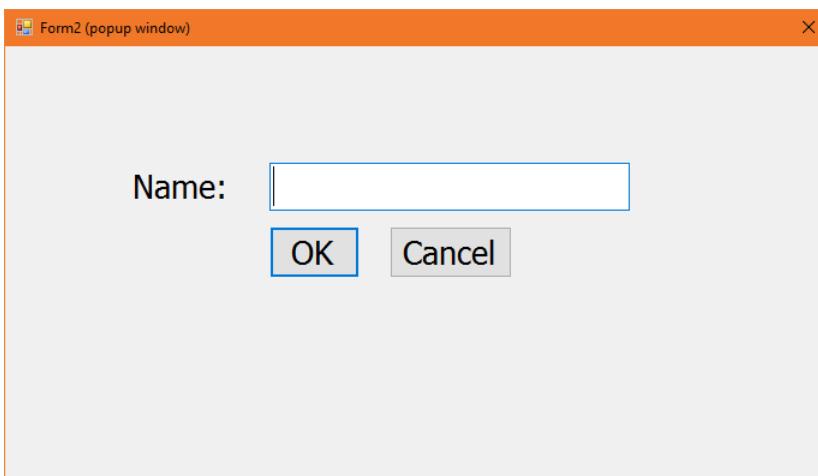
To display popup boxes:

- Create a new windows forms application.
- It automatically creates “Form1”.
- Add “Form2” to the project.
- Set essential properties to the form such as “MinimizeBox”, “MaximizeBox” etc., to the Form2.
- Create object for “Form2” class and call “ShowDialog()” method.
- To exchange data between Form1 and Form2, use a static class called “Common”.

PopupBoxes - Example

Expected Output





Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “PopupBoxesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “PopupBoxesExample”.
- Click on OK.
- Right click on the form and click on “View Code”.
- Open solution explorer. Right click on the project name (PopupBoxesExample) and click on “Add” – “New Item”. Click on “Windows Forms” – “Windows Form”. Enter the form name as “Form2.cs”. Click on “Add”.
- Open solution explorer. Right click on the project name (PopupBoxesExample) and click on “Add” – “New Item”. Click on “Code” – “Class”. Enter the form name as “Common.cs”. Click on “Add”.

Common.cs

using System;

```
namespace PopupWindowsExample
{
    static class Common
    {
        public static string Username { get; set; }
    }
}
```

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PopupWindowsExample
{
    public partial class Form1 : Form
    {
        Button button1;
        Label label1;

        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            /* button1 */
            button1 = new Button();
            button1.AutoSize = true;
            button1.Text = "Open Popup window";
            button1.Location = new Point(100, 100);
            button1.Click += Button1_Click;
            this.Controls.Add(button1);
            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Name appears here";
        }
    }
}
```

```
label1.Location = new Point(100, 300);
this.Controls.Add(label1);
}

private void Button1_Click(object sender, EventArgs e)
{
    Form2 f = new Form2();
    if (f.ShowDialog() == DialogResult.OK)
    {
        label1.Text = Common.Username;
    }
    else
    {
        label1.Text = "";
    }
}
}
```

Form2.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PopupWindowsExample
{
    public partial class Form2 : Form
    {
        Label label1;
        TextBox textbox1;
        Button button1, button2;

        public Form2()
        {
            InitializeComponent();

            /* form properties */

```

```
this.Font = new Font("Tahoma", 20);
this.Size = new Size(700, 400);
this.Text = "Form2 (popup window)";
this.StartPosition = FormStartPosition.CenterScreen;
this.FormBorderStyle = FormBorderStyle.FixedSingle;
this.ShowInTaskbar = false;
this.MinimizeBox = false;
this.MaximizeBox = false;

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Name: ";
label1.Location = new Point(100, 100);
this.Controls.Add(label1);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 30);
textbox1.Location = new Point(220, 97);
this.Controls.Add(textbox1);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "OK";
button1.Location = new Point(220, 150);
button1.Click += Button1_Click;
button1.DialogResult = DialogResult.OK;
this.Controls.Add(button1);
this.AcceptButton = button1;

/* button2 */
button2 = new Button();
button2.AutoSize = true;
button2.Text = "Cancel";
button2.Location = new Point(320, 150);
```

```
button2.Click += Button2_Click;
button2.DialogResult = DialogResult.Cancel;
this.Controls.Add(button2);
this.CancelButton = button2;
}

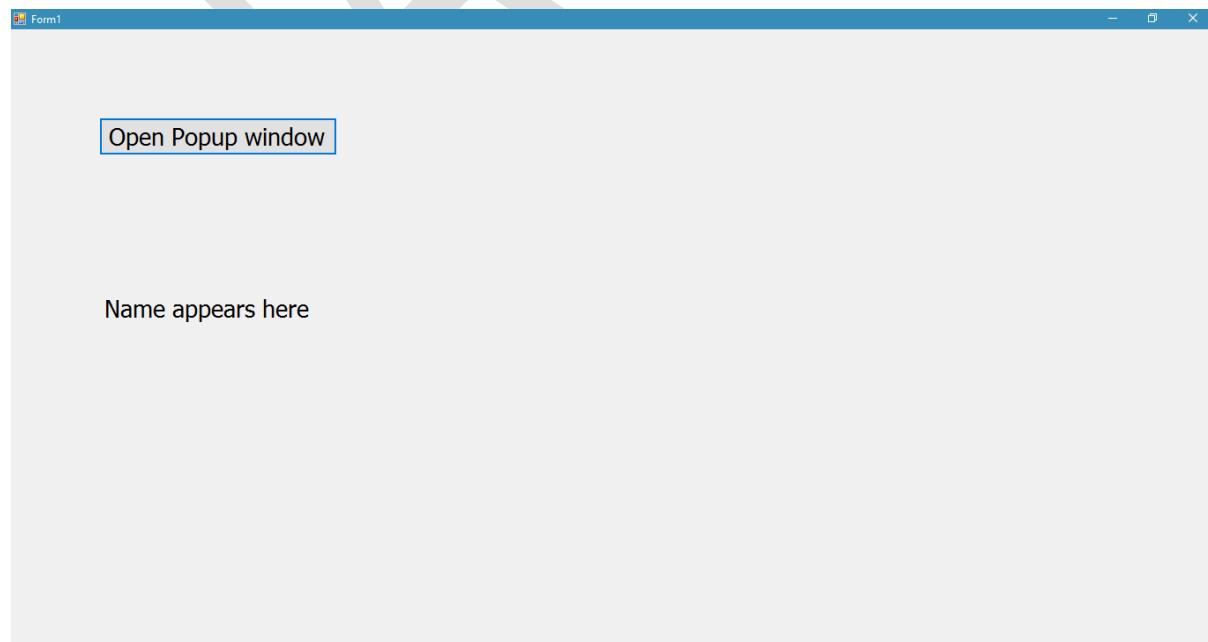
private void Button1_Click(object sender, EventArgs e)
{
    Common.Username = textBox1.Text;
    this.Close();
}

private void Button2_Click(object sender, EventArgs e)
{
    this.Close();
}
}
```

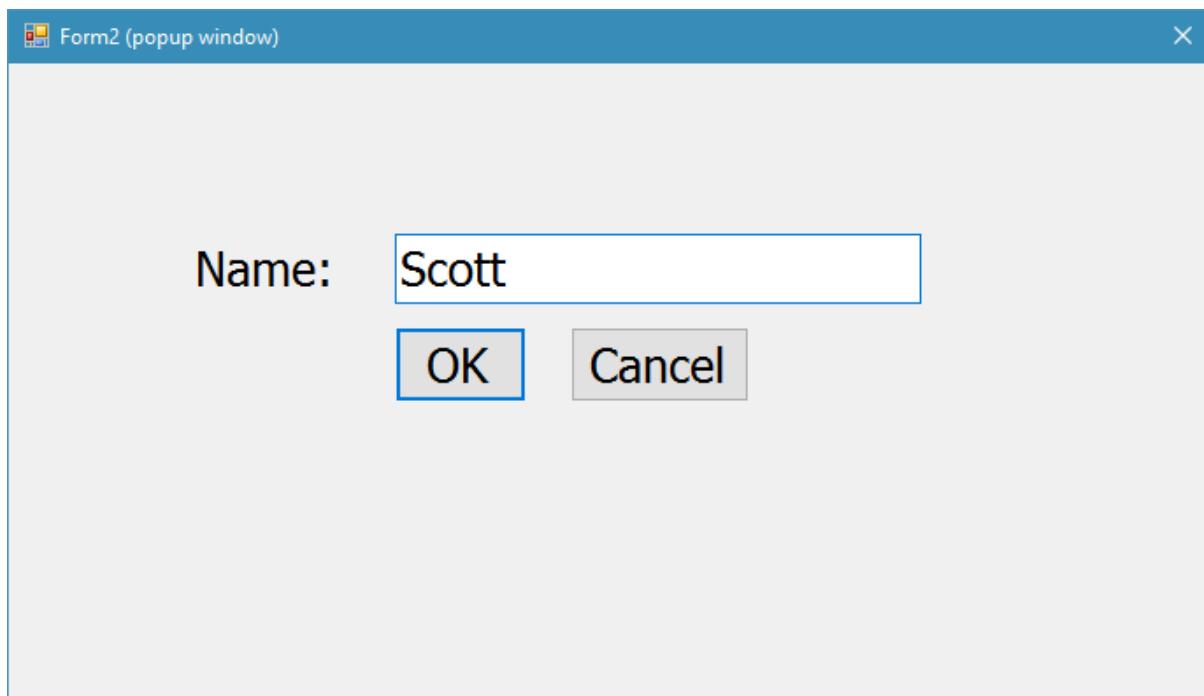
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

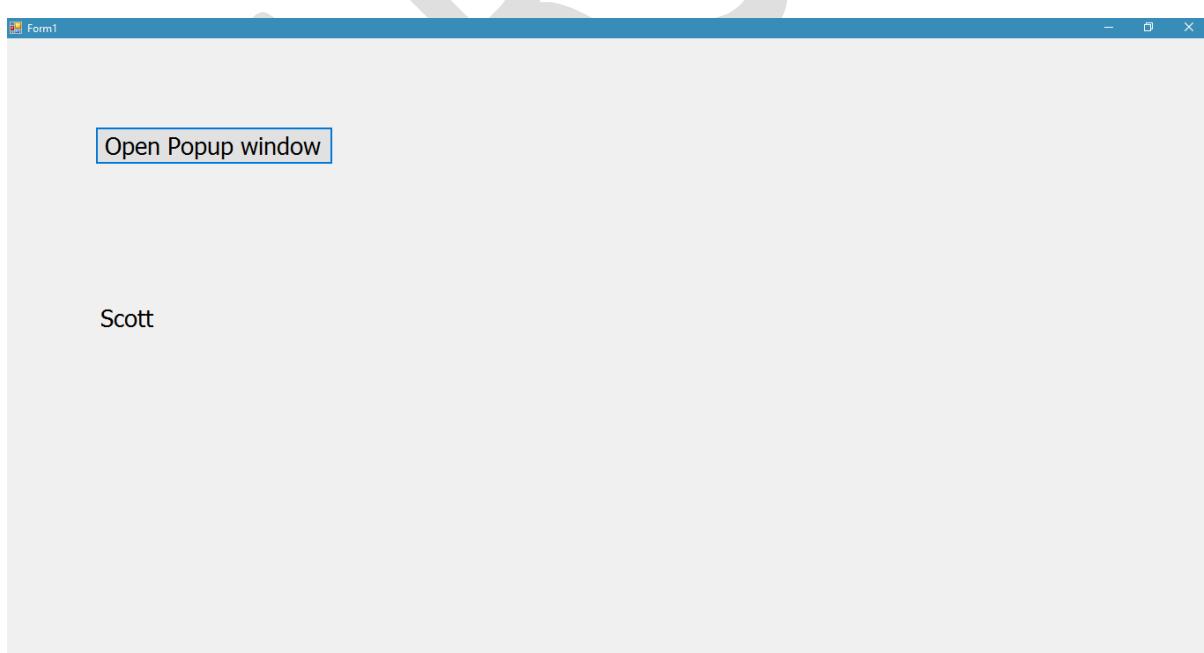
Output



Click on “Open Popup Window”.



Type the any name and click on OK.



“System.Windows.Forms. ColorDialog” class

The “System.Windows.Forms.ColorDialog” class

- “ColorDialog” is a class; “System.Windows.Forms” is a namespace.
- The “ColorDialog” class / control is used to display a color dialogbox (popupbox) on the form, which allows the user to select a color.
- The look and feel of the color dialog is pre-defined.

Steps for development of ColorDialog:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `ColorDialog referencevariable;`
- Create an object:
 - `referencevariable = new ColorDialog();`
- Set properties:
 - `referencevariable.property = value;`
- Call methods:
 - `referencevariable.method();`

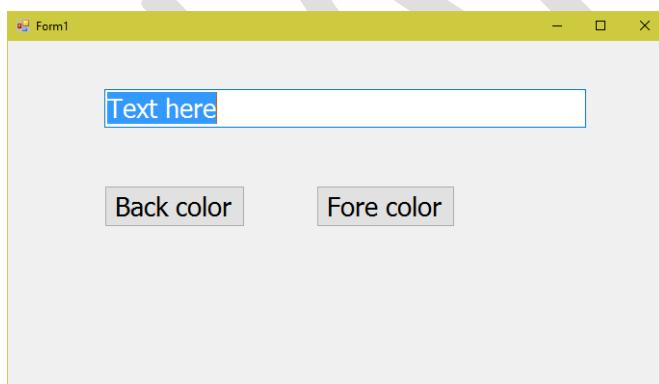
Properties of “ColorDialog” class:

Sl. No	Property	Description

1	Color	Represents currently selected color of the color dialogbox. <u>Syntax:</u> <code>referencevariable.Color = System.Drawing.Color.coloname;</code>
---	-------	---

Methods of "ColorDialog" class:

Sl. No	Method	Description
1	ShowDialog()	Displays the color dialogbox on the form. It returns the clicked button name (OK or Cancel) as "DialogResult". <u>Syntax:</u> <code>referencevariable.ShowDialog()</code>
2	Reset()	Resets all the settings of color dialog. <u>Syntax:</u> <code>referencevariable.Reset();</code>

"System.Windows.Forms. ColorDialog" class - Example**Expected Output**



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ColorDialogExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ColorDialogExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace ColorDialogExample
{
```

```
public partial class Form1 : Form
{
    TextBox textbox1;
    Button button1, button2;
    ColorDialog colordialog1;

    public Form1()
    {
        InitializeComponent();

        /* form properties */
        this.Font = new Font("Tahoma", 20);
        this.Size = new Size(700, 400);

        /* textbox1 */
        textbox1 = new TextBox();
        textbox1.Text = "Text here";
        textbox1.Size = new Size(500, 30);
        textbox1.Location = new Point(100, 50);
        this.Controls.Add(textbox1);

        /* button1 */
        button1 = new Button();
        button1.AutoSize = true;
        button1.Text = "Back color";
        button1.Location = new Point(100, 150);
        button1.Click += Button1_Click;
        this.Controls.Add(button1);

        /* button2 */
        button2 = new Button();
        button2.AutoSize = true;
        button2.Text = "Fore color";
        button2.Location = new Point(320, 150);
        button2.Click += Button2_Click;
        this.Controls.Add(button2);
    }
}
```

```
/* colordialog1 */
colordialog1 = new ColorDialog();
}

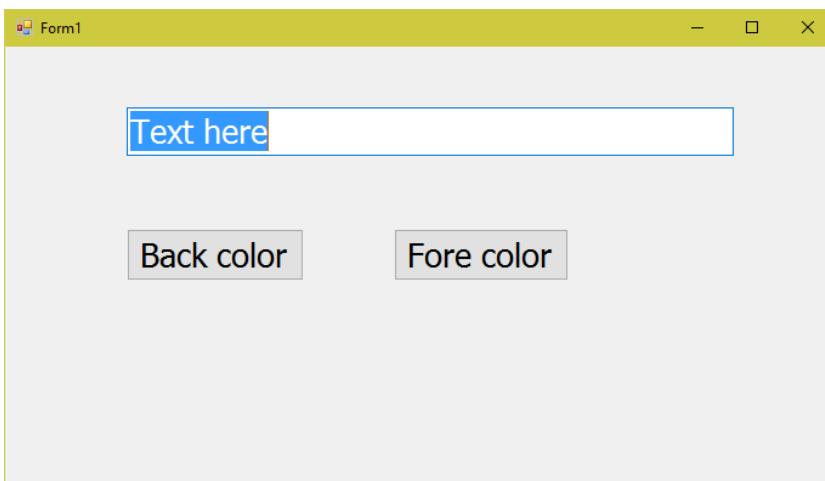
private void Button1_Click(object sender, EventArgs e)
{
    colordialog1.Color = textBox1.BackColor;
    if (colordialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.BackColor = colordialog1.Color;
    }
}

private void Button2_Click(object sender, EventArgs e)
{
    colordialog1.Color = textBox1.ForeColor;
    if (colordialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.ForeColor = colordialog1.Color;
    }
}
```

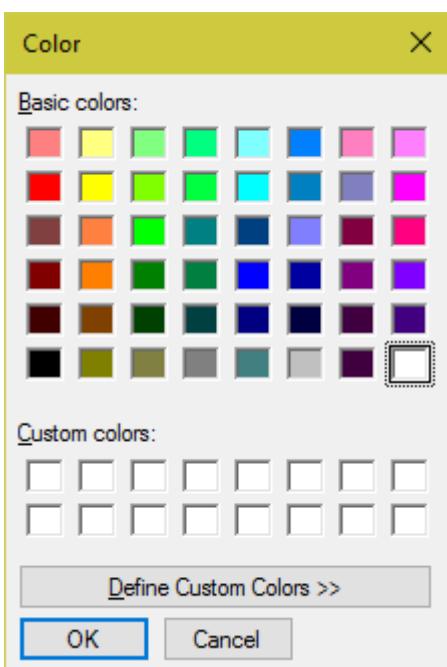
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Back color” and “Fore Color”.



“System.Windows.Forms.FontDialog” class

The “System.Windows.Forms.FontDialog” class

- “FontDialog” is a class; “System.Windows.Forms” is a namespace.
- The “FontDialog” class / control is used to display a font dialogbox (popupbox) on the form, which allows the user to select a font.
- The look and feel of the font dialog is pre-defined.

Steps for development of FontDialog:

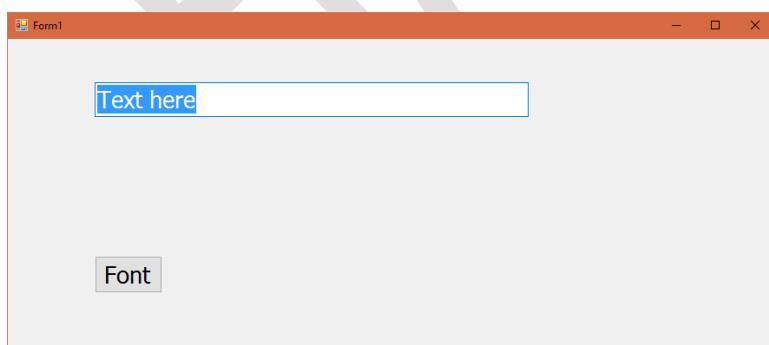
- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `FontDialog referencevariable;`
- Create an object:
 - `referencevariable = new FontDialog();`
- Set properties:
 - `referencevariable.property = value;`
- Call methods:
 - `referencevariable.method();`

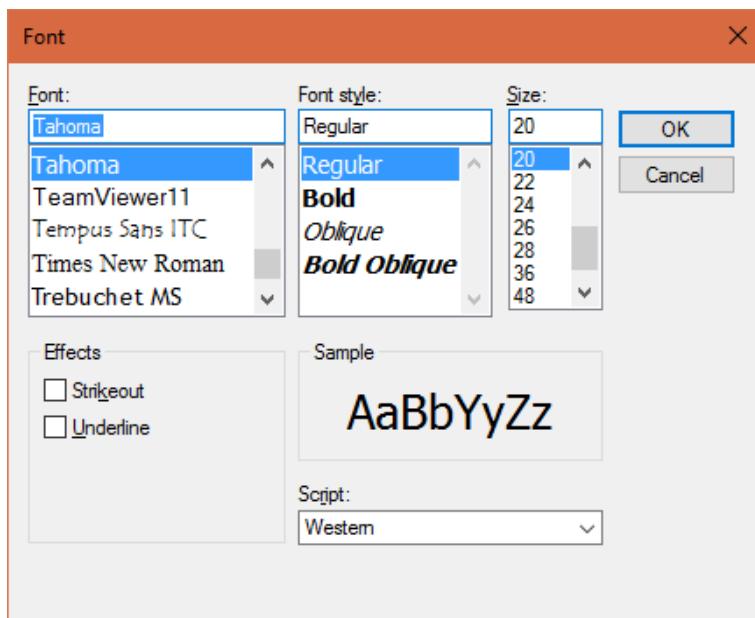
Properties of “FontDialog” class:

Sl. No	Property	Description
1	Font	Represents currently selected font of the font dialogbox. <u>Syntax:</u> <i>referencevariable.Font = new System.Drawing.Font("Font name", font size);</i>

Methods of “FontDialog” class:

Sl. No	Method	Description
1	ShowDialog()	Displays the font dialogbox on the form. It returns the clicked button name (OK or Cancel) as “DialogResult”. <u>Syntax:</u> <i>referencevariable.ShowDialog()</i>
2	Reset()	Resets all the settings of font dialog. <u>Syntax:</u> <i>referencevariable.Reset();</i>

“System.Windows.Forms. FontDialog” class - Example**Expected Output**



Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FontDialogExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FontDialogExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace FontDialogExample
{
    public partial class Form1 : Form
    {

```

```
TextBox textbox1;
Button button1;
FontDialog fontdialog1;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(900, 400);

    /* textbox1 */
    textbox1 = new TextBox();
    textbox1.Text = "Text here";
    textbox1.Size = new Size(500, 30);
    textbox1.Location = new Point(100, 50);
    this.Controls.Add(textbox1);

    /* button1 */
    button1 = new Button();
    button1.AutoSize = true;
    button1.Text = "Font";
    button1.Location = new Point(100, 250);
    button1.Click += Button1_Click;
    this.Controls.Add(button1);

    /* fontdialog1 */
    fontdialog1 = new FontDialog();
}

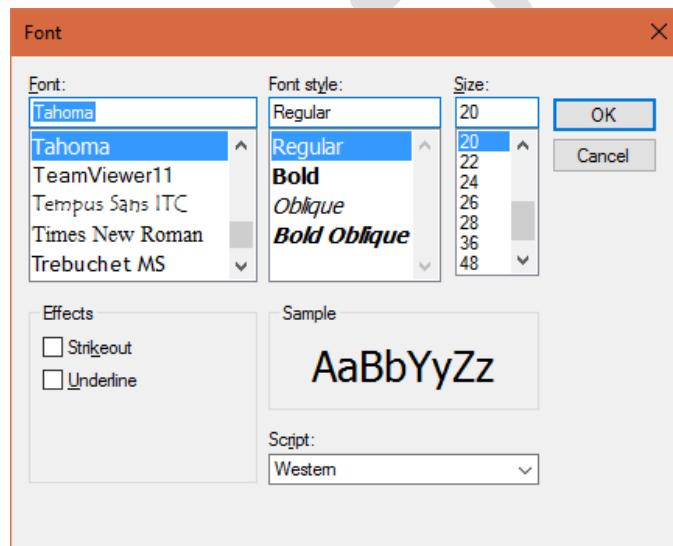
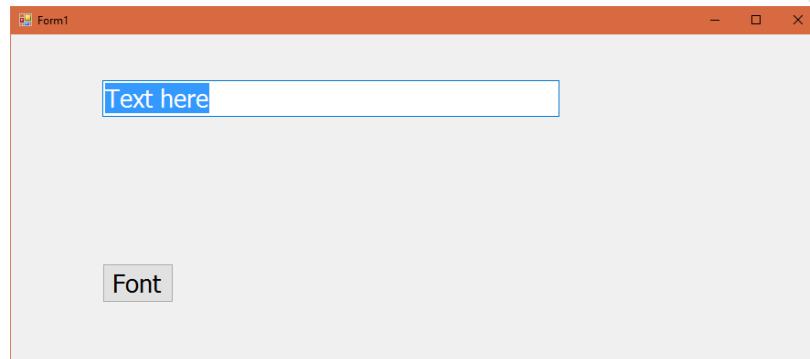
private void Button1_Click(object sender, EventArgs e)
{
    fontdialog1.Font = textbox1.Font;
    if (fontdialog1.ShowDialog() == DialogResult.OK)
    {
        textbox1.Font = fontdialog1.Font;
    }
}
```

```
    }  
}  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Font”, select any font and click on OK.

“System.Windows.Forms.FolderBrowserDialog” class

- “FolderBrowserDialog” is a class; “System.Windows.Forms” is a namespace.

- The “FolderBrowserDialog” class / control is used to display a folder browser dialogbox (popupbox) on the form, which allows the user to select a folder.
- The look and feel of the folder browser dialog is pre-defined.

Steps for development of FolderBrowserDialog:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `FolderBrowserDialog referencevariable;`
- Create an object:
 - `referencevariable = new FolderBrowserDialog();`
- Set properties:
 - `referencevariable.property = value;`
- Call methods:
 - `referencevariable.method();`

Properties of “FolderBrowserDialog” class:

Sl. No	Property	Description
1	SelectedPath	Represents currently selected path of the folder browser dialogbox. <u>Syntax:</u> <code><i>referencevariable.SelectedPath</i> = “any folder path”;</code>

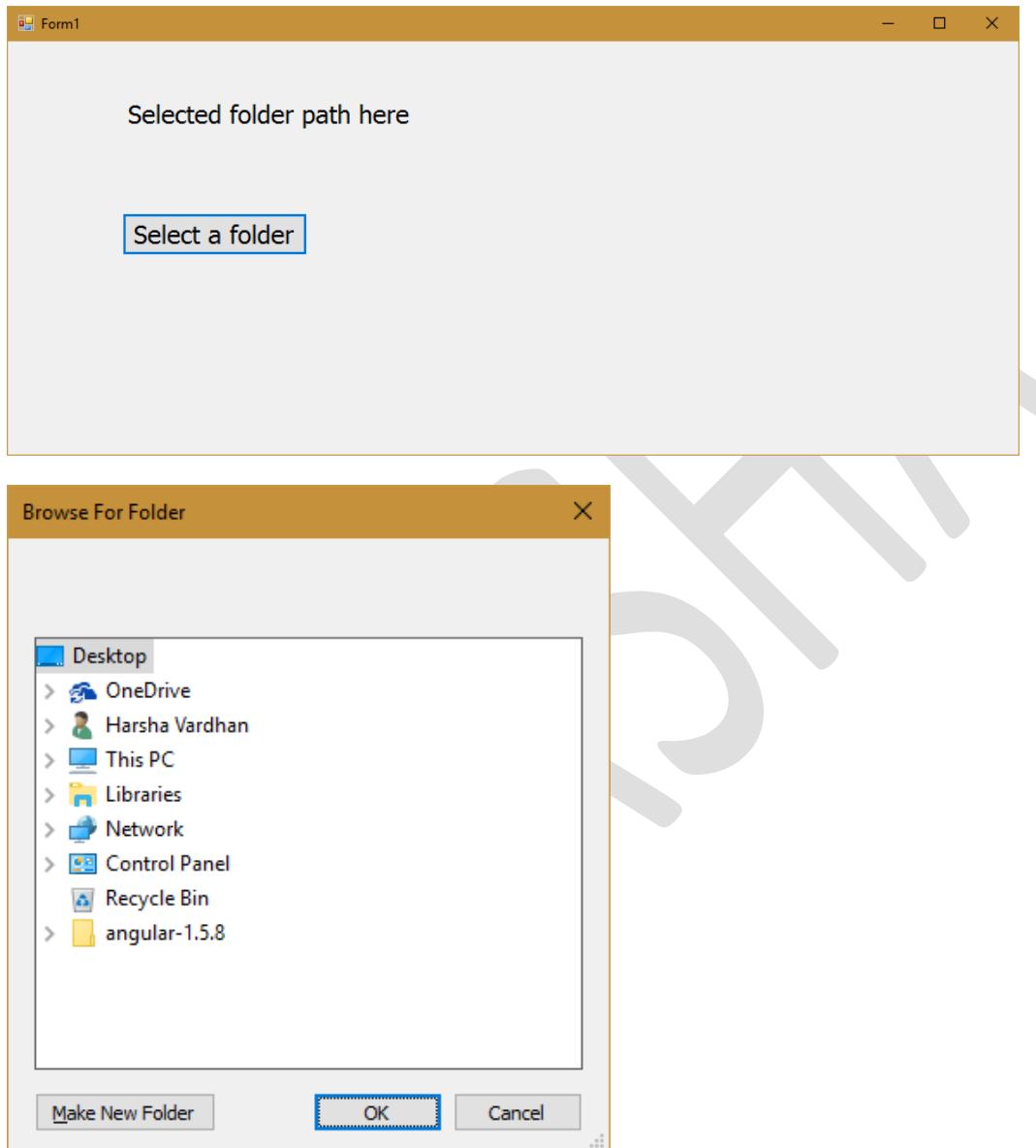
Methods of "FolderBrowserDialog" class:

Sl. No	Method	Description
1	ShowDialog()	Displays the folder browser dialogbox on the form. It returns the clicked button name (OK or Cancel) as "DialogResult". <u>Syntax:</u> <i>referencevariable.ShowDialog()</i>
2	Reset()	Resets all the settings of folder browser dialog. <u>Syntax:</u> <i>referencevariable.Reset();</i>

“System.Windows.Forms.FolderBrowserDialog” class

- Example

Expected Output



Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FolderBrowserDialogExample”.

- Type the location as “C:\CSharp”.
- Type the solution name as “FolderBrowserDialogExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FolderBrowserDialogExample
{
    public partial class Form1 : Form
    {
        Label label1;
        Button button1;
        FolderBrowserDialog folderbrowserdialog1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 16);
            this.Size = new Size(900, 400);

            /* textbox1 */
            label1 = new Label();
            label1.Text = "Selected folder path here";
            label1.AutoSize = true;
            label1.Location = new Point(100, 50);
            this.Controls.Add(label1);

            /* button1 */
            button1 = new Button();
            button1.AutoSize = true;
```

```
button1.Text = "Select a folder";
button1.Location = new Point(100, 150);
button1.Click += Button1_Click;
this.Controls.Add(button1);

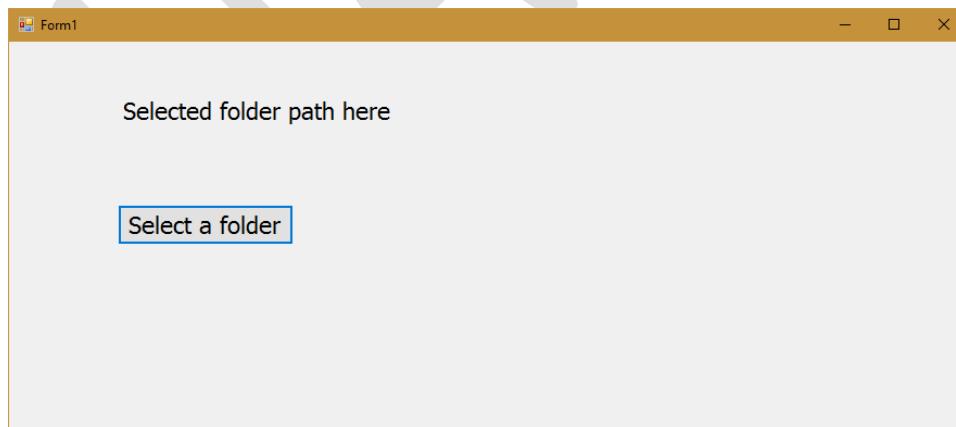
/* folderbrowserdialog1 */
folderbrowserdialog1 = new FolderBrowserDialog();
}

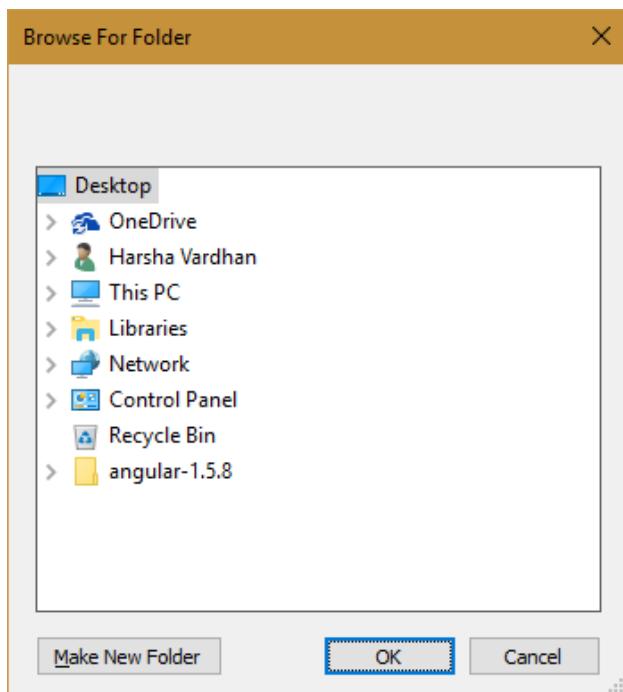
private void Button1_Click(object sender, EventArgs e)
{
    if (folderbrowserdialog1.ShowDialog() == DialogResult.OK)
    {
        label1.Text = folderbrowserdialog1.SelectedPath;
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output





Click on “Select a folder” and select any folder and click on OK.

“System.Windows.Forms. OpenFileDialog” class

The “System.Windows.Forms.OpenFileDialog” class

- “OpenFileDialog” is a class; “System.Windows.Forms” is a namespace.
- The “OpenFileDialog” class / control is used to display a open file dialogbox (popupbox) on the form, which allows the user to select an existing file.
- The look and feel of the open file dialog is pre-defined.

Steps for development of OpenFileDialog:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:

- `OpenFileDialog referencevariable;`

- Create an object:
 - `referencevariable = new OpenFileDialog();`

- Set properties:
 - `referencevariable.property = value;`

- Call methods:
 - `referencevariable.method();`

Properties of “`OpenFileDialog`” class:

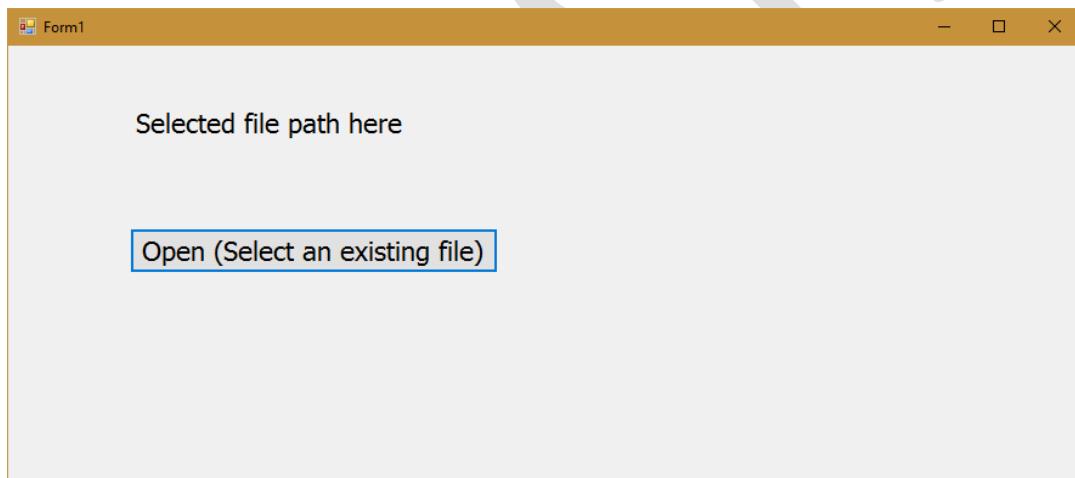
Sl. No	Property	Description
1	<code>FileName</code>	Represents currently selected file name of the open file dialogbox. <u>Syntax:</u> <code>referencevariable.FileName = "any file path";</code>
2	<code>Filenames</code>	Represents currently selected file names of the open file dialogbox. <u>Syntax:</u> <code>referencevariable.Filenames = "any file path";</code>
3	<code>Filter</code>	Represents the filters list of the open file dialogbox. <u>Syntax:</u> <code>referencevariable.Filter = "any filter here";</code>
4	<code>MultiSelect</code>	True/False <u>True:</u> The user can select multiple files. <u>False:</u> The user can select only one file.

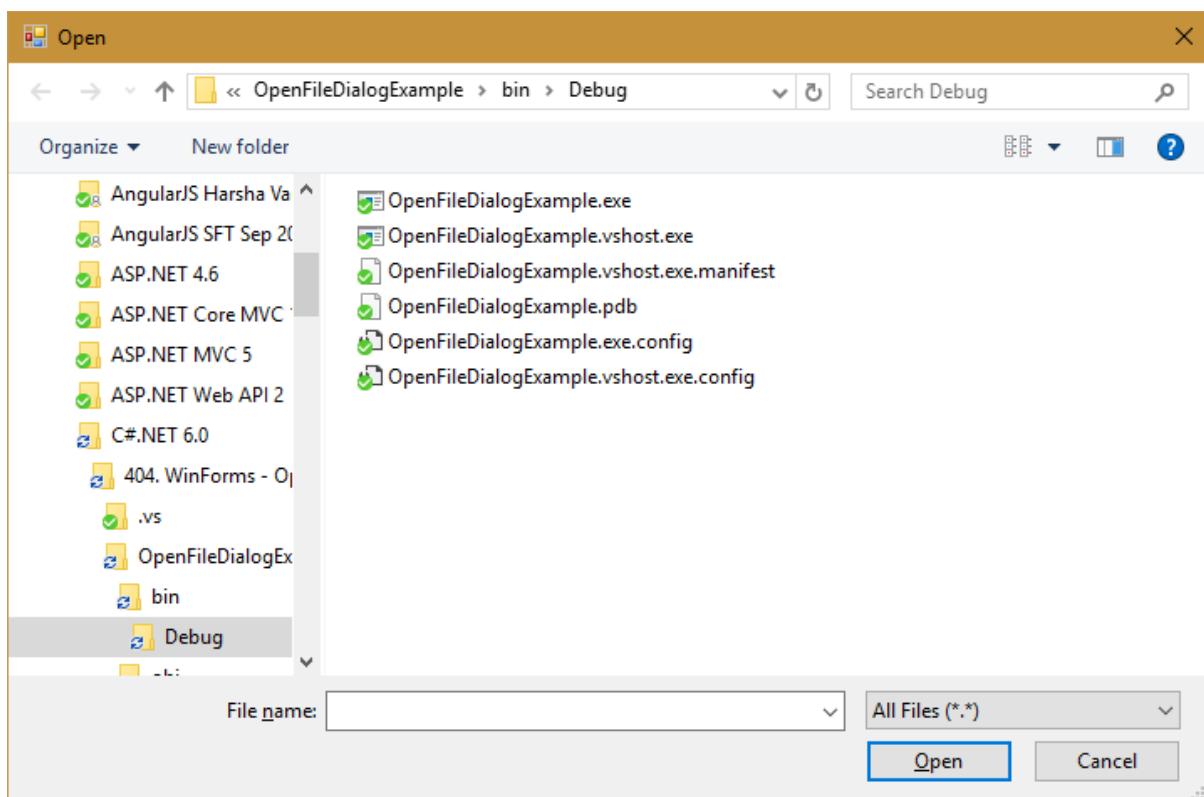
Methods of “`OpenFileDialog`” class:

Sl. No	Method	Description
1	ShowDialog()	Displays the open file dialogbox on the form. It returns the clicked button name (OK or Cancel) as “DialogResult”. <u>Syntax:</u> <i>referencevariable.ShowDialog()</i>
2	Reset()	Resets all the settings of open file dialog. <u>Syntax:</u> <i>referencevariable.Reset();</i>

“System.Windows.Forms.OpenFileDialog” class - Example

Expected Output





Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ OpenFileDialogExample ”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ OpenFileDialogExample ”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;  
using System.Drawing;
```

```
using System.Windows.Forms;
```

```
namespace OpenFileDialogExample
```

```
{
```

```
    public partial class Form1 : Form
```

```
{
```

```
    Label label1;
```

```
    Button button1;
```

```
    OpenFileDialog openFileDialog1;
```

```
    public Form1()
```

```
{
```

```
    InitializeComponent();
```

```
    /* form properties */
```

```
    this.Font = new Font("Tahoma", 16);
```

```
    this.Size = new Size(900, 400);
```

```
    /* textbox1 */
```

```
    label1 = new Label();
```

```
    label1.Text = "Selected file path here";
```

```
    label1.AutoSize = true;
```

```
    label1.Location = new Point(100, 50);
```

```
    this.Controls.Add(label1);
```

```
    /* button1 */
```

```
    button1 = new Button();
```

```
    button1.AutoSize = true;
```

```
    button1.Text = "Open (Select an existing file)";
```

```
    button1.Location = new Point(100, 150);
```

```
    button1.Click += Button1_Click;
```

```
    this.Controls.Add(button1);
```

```
    /* openFileDialog1 */
```

```
    openFileDialog1 = new OpenFileDialog();
```

```
    openFileDialog1.Multiselect = false;
```

```
    openFileDialog1.Filter = "All Files|*.*|Text Files|*.txt";
```

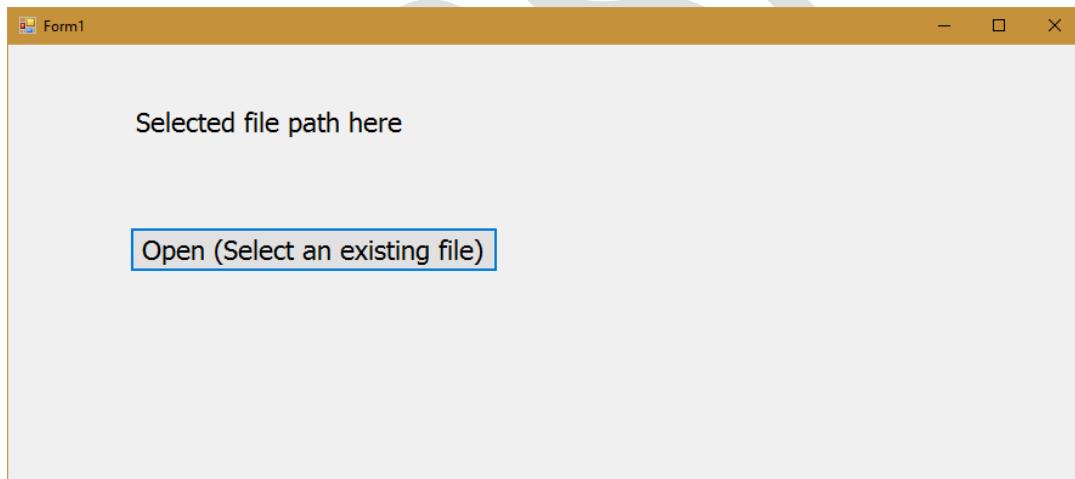
```
}

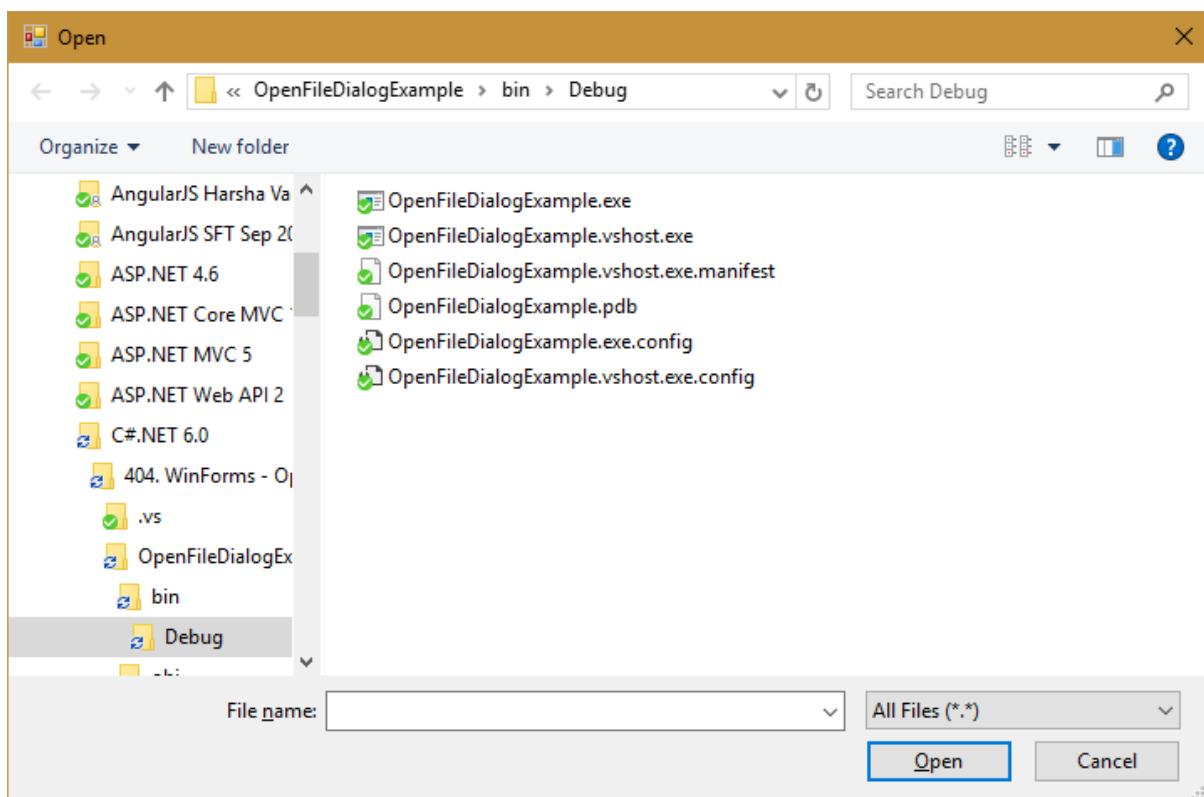
private void Button1_Click(object sender, EventArgs e)
{
    if (openfiledialog1.ShowDialog() == DialogResult.OK)
    {
        label1.Text = openfiledialog1.FileName;
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output





Click on “Open (Select an existing file)”.

Select any existing file and click on Open.

Selected file name appears in the label.

“System.Windows.Forms. SaveFileDialog” class

The “System.Windows.Forms.SaveFileDialog” class

- “SaveFileDialog” is a class; “System.Windows.Forms” is a namespace.
- The “SaveFileDialog” class / control is used to display a save file dialogbox (popupbox) on the form, which allows the user to select a new file.
- The look and feel of the save file dialog is pre-defined.

Steps for development of SaveFileDialog:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `SaveFileDialog referencevariable,`
- Create an object:
 - `referencevariable = new SaveFileDialog();`
- Set properties:
 - `referencevariable.property = value;`
- Call methods:
 - `referencevariable.method();`

Properties of "SaveFileDialog" class:

Sl. No	Property	Description
1	FileName	Represents currently selected file name of the save file dialogbox. <u>Syntax:</u> <code>referencevariable.FileName = "any file path";</code>
2	Filter	Represents the filters list of the save file dialogbox. <u>Syntax:</u> <code>referencevariable.Filter = "any filter here";</code>

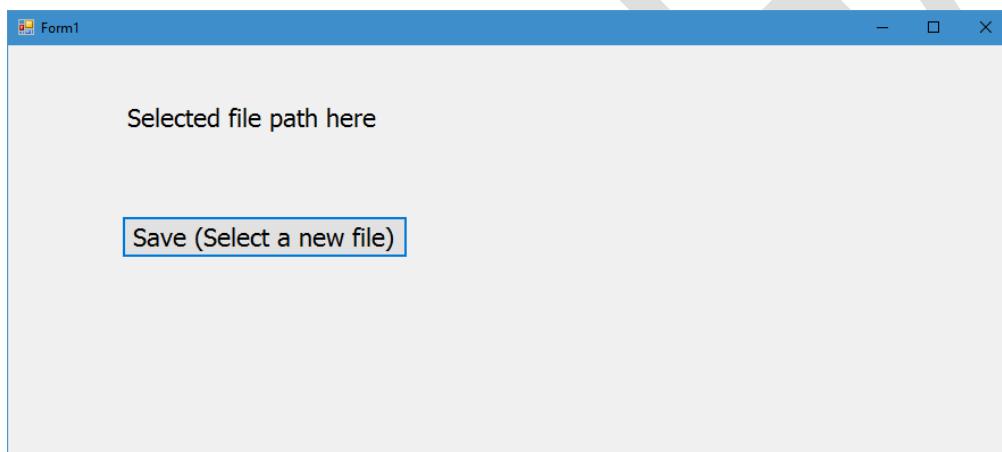
Methods of "SaveFileDialog" class:

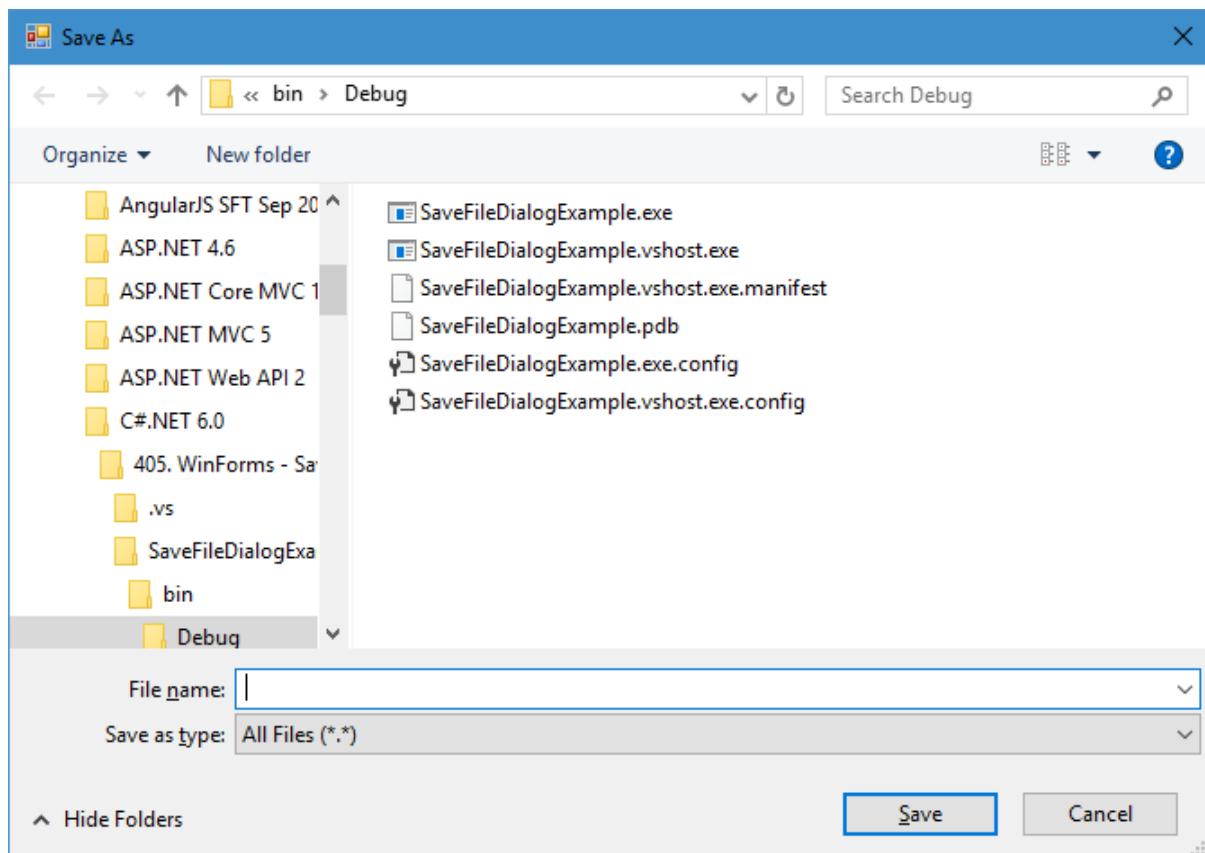
Sl. No	Method	Description

1	ShowDialog()	Displays the save file dialogbox on the form. It returns the clicked button name (OK or Cancel) as “DialogResult”. <u>Syntax:</u> <i>referencevariable.ShowDialog()</i>
2	Reset()	Resets all the settings of save file dialog. <u>Syntax:</u> <i>referencevariable.Reset();</i>

“System.Windows.Forms. SaveFileDialog” class - Example

Expected Output





Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SaveFileDialogExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SaveFileDialogExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace SaveFileDialogExample
{
    public partial class Form1 : Form
    {
        Label label1;
        Button button1;
        SaveFileDialog savefiledialog1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 16);
            this.Size = new Size(900, 400);

            /* textbox1 */
        }
    }
}
```

```
label1 = new Label();
label1.Text = "Selected file path here";
label1.AutoSize = true;
label1.Location = new Point(100, 50);
this.Controls.Add(label1);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Save (Select a new file)";
button1.Location = new Point(100, 150);
button1.Click += Button1_Click;
this.Controls.Add(button1);

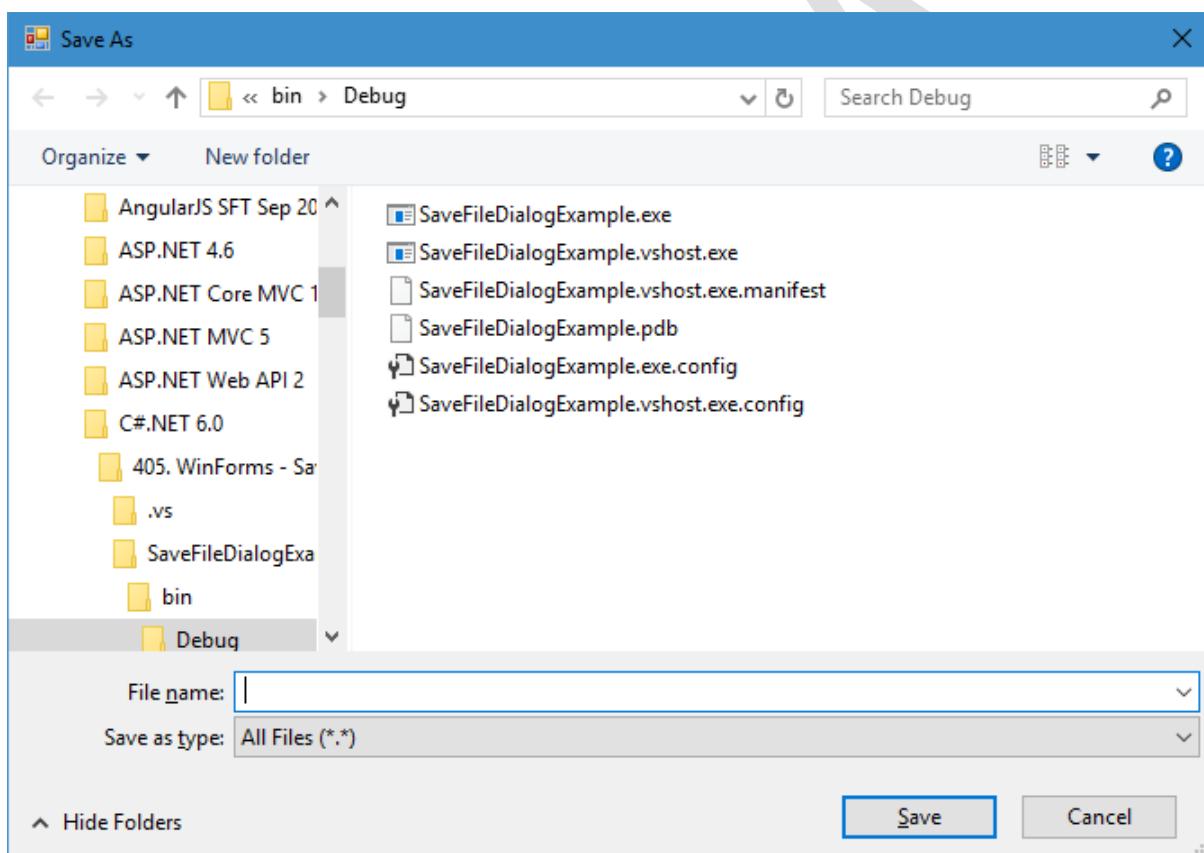
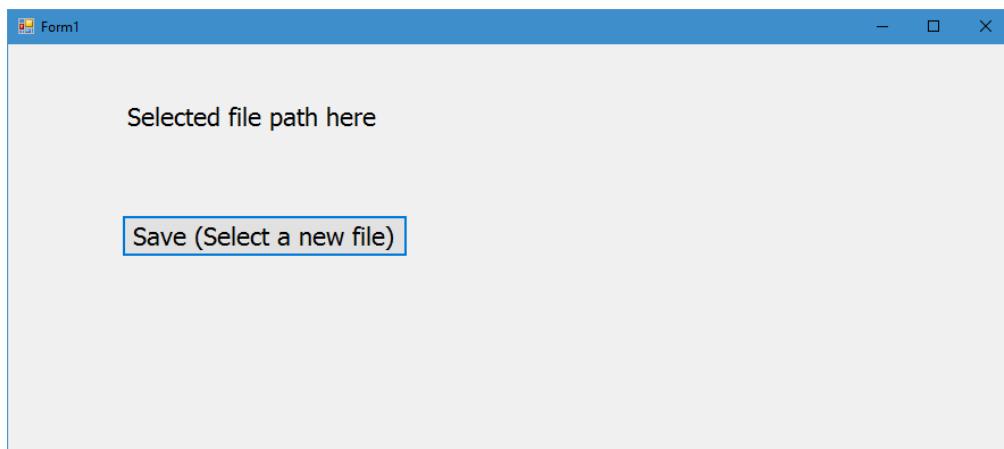
/* savefiledialog1 */
savefiledialog1 = new SaveFileDialog();
savefiledialog1.Filter = "All Files|.*|Text Files|*.txt";
}

private void Button1_Click(object sender, EventArgs e)
{
    if (savefiledialog1.ShowDialog() == DialogResult.OK)
    {
        label1.Text = savefiledialog1.FileName;
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on "Save (Select a new file)".

Type any new file name and click on "Save". The specified file name appears in the label.

"System.Windows.Forms.MenuStrip" class

The "System.Windows.Forms.MenuStrip" class

- “MenuStrip” is a class; “System.Windows.Forms” is a namespace.
- The “MenuStrip” class / control is used to display a menu bar at the top of the form.
- A menu can contain menu items, textboxes, buttons etc.

Steps for development of MenuStrip:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `MenuStrip referencevariable,`
- Create an object:
 - `referencevariable = new ToolStrip();`
- Set properties:
 - `referencevariable.property = value;`
- Call methods:
 - `referencevariable.method();`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “MenuStrip” class:

Sl. No	Property	Description
1	Dock	<p>Displays the control at most top / left / right / bottom side of the form.</p> <p><u>Syntax:</u> <code>referencevariable.Dock = System.Windows.Forms.DockStyle.Optionhere;</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Items	<p>Represents the parent menu items in the menubar.</p> <p><u>Syntax:</u> <code>referencevariable.Items.Add(toolstrip menu item here);</code></p>
5	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
6	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
7	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>

8	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.ForeColor = System.Drawing.Color.Green;</i>
9	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
10	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
11	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>

Events of "MenuStrip" class:

Sl. No	Event	Description
1	ItemClicked	Executes when the user clicks any item in the menubar. <u>Syntax:</u> <i>referencevariable.ItemClicked += methodname;</i>

Properties of "ToolStripMenuItem" class:

The "ToolStripMenuItem" class's object represents a menu item in the menubar.

Sl. No	Property	Description
1	Text	Represents text of the menu item. <u>Syntax:</u> <i>referencevariable.Text = "string here";</i>

2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	DropDownItems	<p>Represents the child menu items in the menu item.</p> <p><u>Syntax:</u> <code>referencevariable.DropDownItems.Add(toolstrip menu item here);</code></p>
5	ShortCutKeys	<p>Represents shortcut keys of the menu item.</p> <p><u>Options:</u> All alphabets, digits, special keys, function keys.</p> <p><u>Syntax:</u> <code>referencevariable.ShortCutKeys = Keys.your option here;</code></p>
6	Image	<p>Represents icon image of the menu item.</p> <p><u>Syntax:</u> <code>referencevariable.Image = System.Drawing.Image.FromFile("image file path here");</code></p>
7	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>
8	ForeColor	<p>Represents foreground color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code></p>
9	Visible	<p><u>True:</u> The control appears on the form.</p> <p><u>False:</u> The control will disappear on the form.</p> <p><u>Syntax:</u> <code>referencevariable.Visible = true false;</code></p>

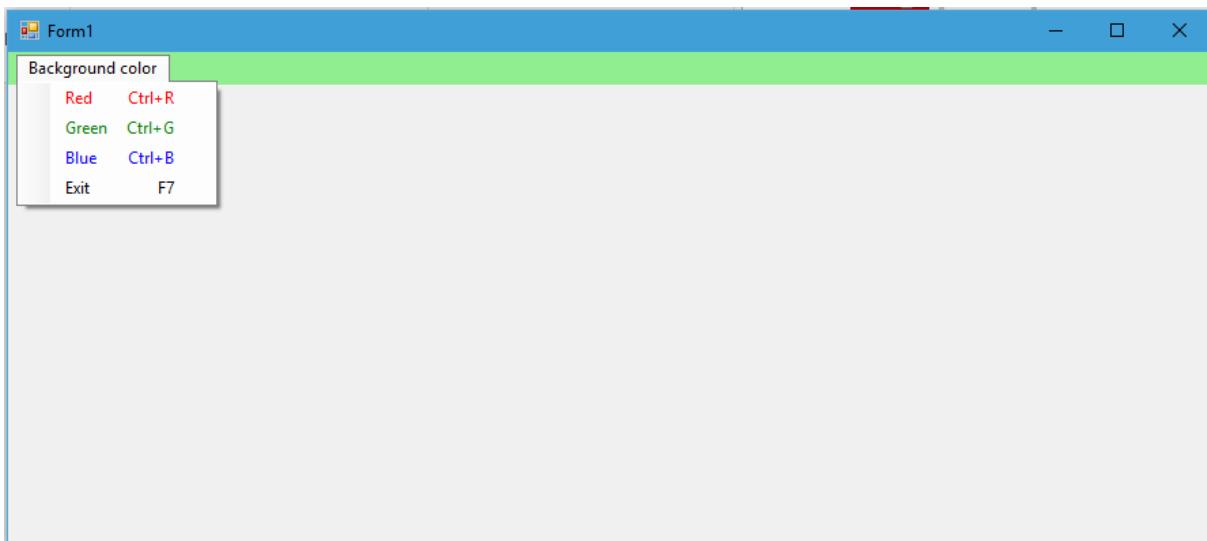
10	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
11	Checked	<p>True / False.</p> <p><u>True:</u> Checkbox of the menu item is checked.</p> <p><u>False:</u> Checkbox of the menu item is unchecked.</p> <p><u>Syntax:</u> <code>referencevariable.Checked = true false;</code></p>

Events of "ToolStripMenuItem" class:

Sl. No	Event	Description
1	Click	<p>Executes when the user clicks on the menu item.</p> <p><u>Syntax:</u> <code>referencevariable.Click += methodname;</code></p>
2	MouseEnter	<p>Executes when the user places the mouse pointer on the menu item.</p> <p><u>Syntax:</u> <code>referencevariable.Click += methodname;</code></p>
3	MouseLeave	<p>Executes when the user moves the mouse pointer out of the menu item.</p> <p><u>Syntax:</u> <code>referencevariable.MouseLeave += methodname;</code></p>

"System.Windows.Forms.MenuStrip" class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MenuStripExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MenuStripExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace MenuStripExample
{
    public partial class Form1 : Form
```

```
{  
    MenuStrip menustrip1;  
    ToolStripMenuItem parentmenuItem1, menuItem1, menuItem2,  
    menuItem3, menuItem4;  
  
    public Form1()  
    {  
        InitializeComponent();  
  
        /* form properties */  
        this.Font = new Font("Tahoma", 20);  
        this.Size = new Size(900, 400);  
  
        /* menustrip1 */  
        menustrip1 = new ToolStrip();  
        menustrip1.Dock = DockStyle.Top;  
        menustrip1.BackColor = Color.LightGreen;  
        this.Controls.Add(menustrip1);  
  
        /* parentmenuItem1 */  
        parentmenuItem1 = new ToolStripMenuItem();  
        parentmenuItem1.Text = "Background color";  
        menustrip1.Items.Add(parentmenuItem1);  
  
        /* menuItem1 */  
        menuItem1 = new ToolStripMenuItem();  
        menuItem1.Text = "Red";  
        menuItem1.ShortcutKeys = (Keys)(Keys.Control | Keys.R);  
        menuItem1.ForeColor = Color.Red;  
        menuItem1.Click += MenuItem1_Click;  
        parentmenuItem1.DropDownItems.Add(menuItem1);  
  
        /* menuItem2 */  
        menuItem2 = new ToolStripMenuItem();  
        menuItem2.Text = "Green";  
        menuItem2.ShortcutKeys = (Keys)(Keys.Control | Keys.G);  
        menuItem2.ForeColor = Color.Green;
```

```
menuItem2.Click += MenuItem2_Click;
parentmenuItem1.DropDownItems.Add(menuItem2);

/* menuItem3 */
menuItem3 = new ToolStripMenuItem();
menuItem3.Text = "Blue";
menuItem3.ShortcutKeys = (Keys)(Keys.Control | Keys.B);
menuItem3.ForeColor = Color.Blue;
menuItem3.Click += MenuItem3_Click;
parentmenuItem1.DropDownItems.Add(menuItem3);

/* menuItem4 */
menuItem4 = new ToolStripMenuItem();
menuItem4.Text = "Exit";
menuItem4.ShortcutKeys = Keys.F7;
menuItem4.Click += MenuItem4_Click;
parentmenuItem1.DropDownItems.Add(menuItem4);
}

private void MenuItem1_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Red;
}

private void MenuItem2_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Green;
}

private void MenuItem3_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Blue;
}

private void MenuItem4_Click(object sender, EventArgs e)
{
    this.Close();
}
```

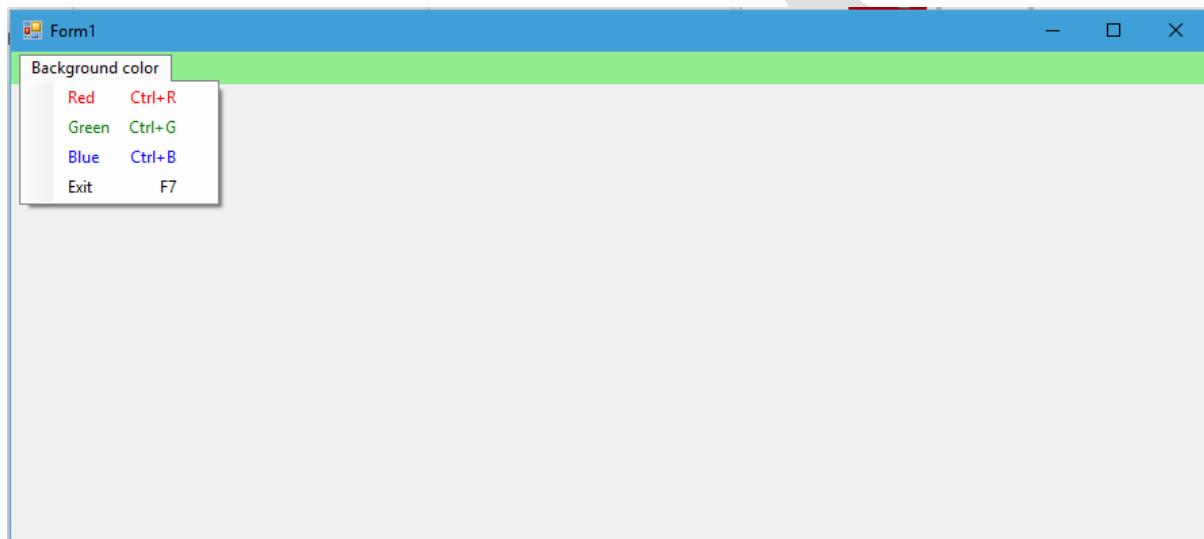
```

    }
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Go to “Background color” menu and click on “Red”, “Green”, “Blue” and “Exit” options.

“System.Windows.Forms.ContextMenuStrip” class

- “ContextMenuStrip” is a class; “System.Windows.Forms” is a namespace.
- The “ContextMenuStrip” class / control is used to display a right click menu for a form or any control.
- A context menu can contain menu items, textboxes, buttons etc.

Steps for development of ContextMenuStrip:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `ContextMenuStrip referencevariable,`
- Create an object:
 - `referencevariable = new ContextMenuStrip();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "ContextMenuStrip" class:

Sl. No	Property	Description
1	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
2	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
3	Items	Represents the parent menu items in the context menu. <u>Syntax:</u> <code>referencevariable.Items.Add(toolstrip menu item here);</code>
4	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code>
5	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
6	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>
7	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>

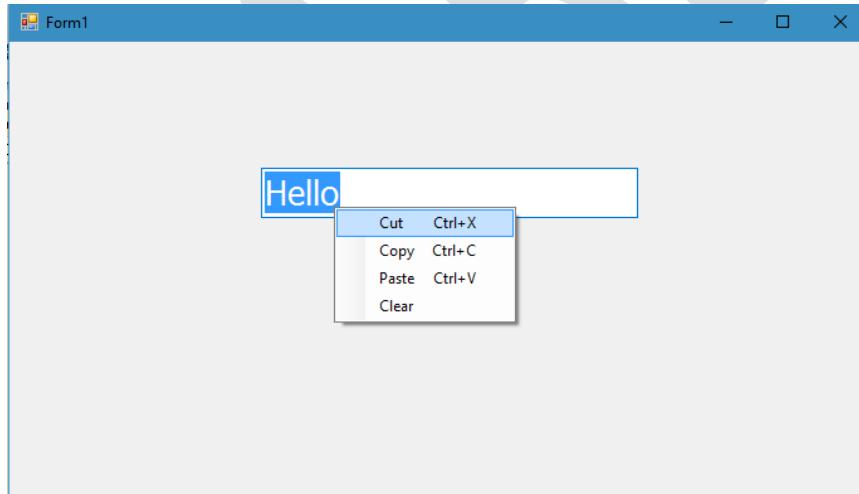
8	Enabled	<p><u>True:</u> The control works.</p> <p><u>False:</u> The control doesn't work. That means, the control can't be used by the user.</p> <p><u>Syntax:</u> <code>referencevariable.Enabled = true false;</code></p>
---	---------	---

Events of “ContextMenuStrip” class:

Sl. No	Event	Description
1	ItemClicked	<p>Executes when the user clicks any item in the context menu.</p> <p><u>Syntax:</u> <code>referencevariable.ItemClicked += methodname;</code></p>

“System.Windows.Forms.ContextMenuStrip” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.

- Type the project name as “ContextMenuStripExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ContextMenuStripExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace ContextMenuStripExample
{
    public partial class Form1 : Form
    {
        TextBox textbox1;
        ContextMenuStrip contextMenuStrip1;
        ToolStripMenuItem menuItem1, menuItem2, menuItem3, menuItem4;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(700, 400);

            /* textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(300, 40);
            textbox1.Location = new Point(200, 100);
            textbox1.Text = "Hello";
            this.Controls.Add(textbox1);

            /* contextMenuStrip1 */
        }
    }
}
```

```
contextmenustrip1 = new ContextMenuStrip();
textbox1.ContextMenuStrip = contextmenustrip1;

/* menuitem1 */
menuitem1 = new ToolStripMenuItem();
menuitem1.Text = "Cut";
menuitem1.ShortcutKeys = (Keys)(Keys.Control | Keys.X);
menuitem1.Click += MenuItem1_Click;
contextmenustrip1.Items.Add(menuitem1);

/* menuitem2 */
menuitem2 = new ToolStripMenuItem();
menuitem2.Text = "Copy";
menuitem2.ShortcutKeys = (Keys)(Keys.Control | Keys.C);
menuitem2.Click += MenuItem2_Click;
contextmenustrip1.Items.Add(menuitem2);

/* menuitem3 */
menuitem3 = new ToolStripMenuItem();
menuitem3.Text = "Paste";
menuitem3.ShortcutKeys = (Keys)(Keys.Control | Keys.V);
menuitem3.Click += MenuItem3_Click;
contextmenustrip1.Items.Add(menuitem3);

/* menuitem4 */
menuitem4 = new ToolStripMenuItem();
menuitem4.Text = "Clear";
menuitem4.Click += MenuItem4_Click;
contextmenustrip1.Items.Add(menuitem4);
}

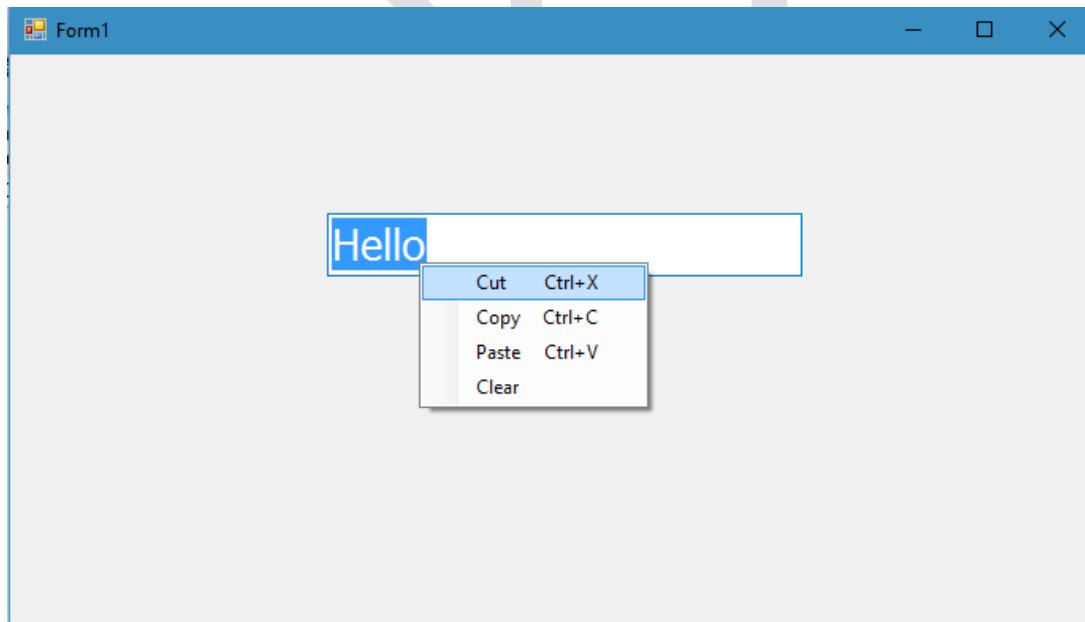
private void MenuItem1_Click(object sender, EventArgs e)
{
    textbox1.Cut();
}
private void MenuItem2_Click(object sender, EventArgs e)
{
```

```
    textbox1.Copy();
}
private void MenuItem3_Click(object sender, EventArgs e)
{
    textbox1.Paste();
}
private void MenuItem4_Click(object sender, EventArgs e)
{
    textbox1.Clear();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Type some text and right click on the textbox. Click on “Cut”, “Copy”, “Paste”, “Clear”.

“System.Windows.Forms.ToolStrip” class

- “ToolStrip” is a class; “System.Windows.Forms” is a namespace.
- The “ToolStrip” class / control is used to display a toolbar at the top of the form.
- A toolbar can contain textboxes, buttons, progress bars etc.

Steps for development of ToolStrip:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `ToolStrip referencevariable;`
- Create an object:
 - `referencevariable = new ToolStrip();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “ToolStrip” class:

Sl. No	Property	Description

1	Dock	Displays the control at most top / left / right / bottom side of the form. <u>Syntax:</u> <i>referencevariable.Dock = System.Windows.Forms.DockStyle.Optionhere;</i>
2	Font	Represents font settings of the control. <u>Syntax:</u> <i>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</i>
3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <i>referencevariable.Size = new System.Drawing.Size(int width, int height);</i>
4	Items	Represents the buttons / textboxes / progressbars in the toolbar. <u>Syntax:</u> <i>referencevariable.Items.Add(toolstrip button here);</i>
5	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <i>referencevariable.Location = new System.Drawing.Point(int x, int y);</i>
6	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <i>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</i>
7	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.BackColor = System.Drawing.Color.Green;</i>
8	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.ForeColor = System.Drawing.Color.Green;</i>

9	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
10	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
11	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>

Events of "ToolStrip" class:

Sl. No	Event	Description
1	ItemClicked	Executes when the user clicks any item in the toolbar. <u>Syntax:</u> <i>referencevariable.ItemClicked += methodname;</i>

Properties of "ToolStripButton" class:

The “ToolStripButton” class's object represents a button in the toolbar.

Sl. No	Property	Description
1	Text	Represents text of the tool strip button. <u>Syntax:</u> <i>referencevariable.Text = "string here";</i>

2	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
4	Image	Represents icon image of the menu item. <u>Syntax:</u> <code>referencevariable.Image = System.Drawing.Image.FromFile("image file path here");</code>
5	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
6	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>
7	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>
8	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <code>referencevariable.Enabled = true false;</code>

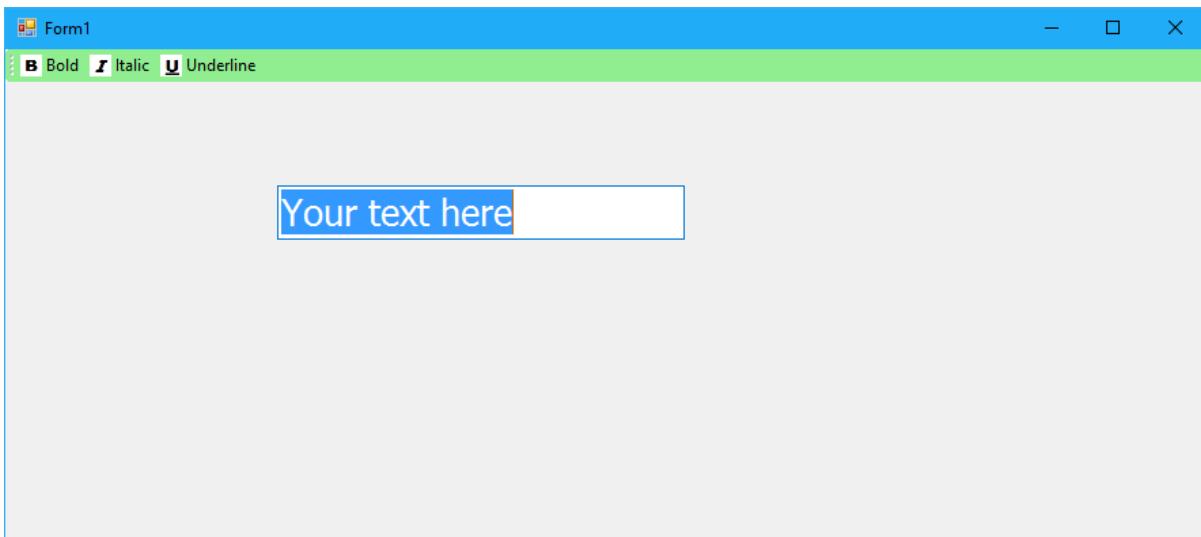
9	Checked	<p>True / False.</p> <p><u>True:</u> Checkbox of the toolStrip button is checked.</p> <p><u>False:</u> Checkbox of the toolStrip button is unchecked.</p> <p><u>Syntax:</u> <code>referencevariable.Checked = true false;</code></p>
---	---------	---

Events of “ToolStripButton” class:

Sl. No	Event	Description
1	Click	<p>Executes when the user clicks on the toolStrip button.</p> <p><u>Syntax:</u> <code>referencevariable.Click += methodname;</code></p>
2	MouseEnter	<p>Executes when the user places the mouse pointer on the toolStrip button.</p> <p><u>Syntax:</u> <code>referencevariable.MouseEnter += methodname;</code></p>
3	MouseLeave	<p>Executes when the user moves the mouse pointer out of the toolStrip button.</p> <p><u>Syntax:</u> <code>referencevariable.MouseLeave += methodname;</code></p>

“System.Windows.Forms.ToolStrip” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ToolStripExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ToolStripExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace ToolStripExample
{
    public partial class Form1 : Form
```

```
{  
    TextBox textbox1;  
    ToolStrip toolStrip1;  
    ToolStripButton menuItem1, menuItem2, menuItem3;  
  
    public Form1()  
    {  
        InitializeComponent();  
  
        /* form properties */  
        this.Font = new Font("Tahoma", 20);  
        this.Size = new Size(900, 400);  
  
        /* textbox1 */  
        textbox1 = new TextBox();  
        textbox1.Size = new Size(300, 40);  
        textbox1.Location = new Point(200, 100);  
        textbox1.Text = "Your text here";  
        this.Controls.Add(textbox1);  
  
        /* toolStrip1 */  
        toolStrip1 = new ToolStrip();  
        toolStrip1.Dock = DockStyle.Top;  
        toolStrip1.BackColor = Color.LightGreen;  
        this.Controls.Add(toolStrip1);  
  
        /* menuItem1 */  
        menuItem1 = new ToolStripButton();  
        menuItem1.Text = "Bold";  
        menuItem1.Image = Image.FromFile(@"C:\CSharp\b.png");  
        menuItem1.Click += MenuItem1_Click;  
        toolStrip1.Items.Add(menuItem1);  
  
        /* menuItem2 */  
        menuItem2 = new ToolStripButton();  
        menuItem2.Text = "Italic";  
        menuItem2.Image = Image.FromFile(@"C:\CSharp\i.png");
```

```
menuItem2.Click += MenuItem2_Click;
toolStrip1.Items.Add(menuItem2);

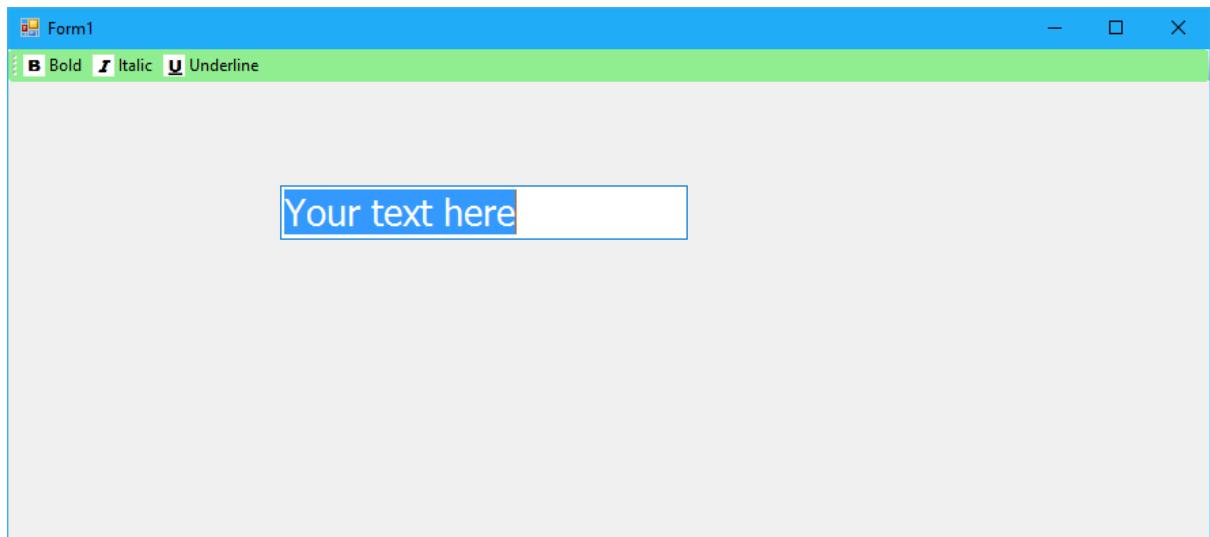
/* menuItem3 */
menuItem3 = new ToolStripButton();
menuItem3.Text = "Underline";
menuItem3.Image = Image.FromFile(@"C:\CSharp\u.png");
menuItem3.Click += MenuItem3_Click;
toolStrip1.Items.Add(menuItem3);
}

private void MenuItem1_Click(object sender, EventArgs e)
{
    textBox1.Font = new Font("Tahoma", 20, FontStyle.Bold);
}
private void MenuItem2_Click(object sender, EventArgs e)
{
    textBox1.Font = new Font("Tahoma", 20, FontStyle.Italic);
}
private void MenuItem3_Click(object sender, EventArgs e)
{
    textBox1.Font = new Font("Tahoma", 20, FontStyle.Underline);
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Try to click on “Bold”, “Italic”, “Underline” options in the toolbar.

HRASH

“System.Windows.Forms.StatusStrip” class

The “System.Windows.Forms.StatusStrip” class

- “StatusStrip” is a class; “System.Windows.Forms” is a namespace.
- The “StatusStrip” class / control is used to display a statusbar at the top of the form.
- A statusbar can contain labels, progress bars etc.

Steps for development of StatusStrip:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `StatusStrip referencevariable,`
- Create an object:
 - `referencevariable = new StatusStrip();`
- Set properties:
 - `referencevariable.property = value;`
- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of “StatusStrip” class:

Sl. No	Property	Description
1	Dock	<p>Displays the control at most top / left / right / bottom side of the form.</p> <p><u>Syntax:</u> <code>referencevariable.Dock = System.Windows.Forms.DockStyle.Optionhere;</code></p>
2	Font	<p>Represents font settings of the control.</p> <p><u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code></p>
3	Size	<p>Represents size (width and height) of the control.</p> <p><u>Options:</u> pixels.</p> <p><u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code></p>
4	Items	<p>Represents the buttons / textboxes / progressbars in the statusbar.</p> <p><u>Syntax:</u> <code>referencevariable.Items.Add(toolstrip status label here);</code></p>
5	Location	<p>Represents position (X and Y) of the control on the form.</p> <p><u>Options:</u> pixels</p> <p><u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code></p>
6	Cursor	<p>Represents mouse cursor of the control.</p> <p><u>Options:</u> Arrow Hand etc.</p> <p><u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code></p>
7	BackColor	<p>Represents background color of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code></p>

8	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <i>referencevariable.ForeColor = System.Drawing.Color.Green;</i>
9	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <i>referencevariable.Visible = true false;</i>
10	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <i>referencevariable.Enabled = true false;</i>
11	TabIndex	Represents tab order. Ex: 1, 2, 3 etc. <u>Syntax:</u> <i>referencevariable.TabIndex = any number;</i>

Events of "StatusStrip" class:

Sl. No	Event	Description
1	ItemClicked	Executes when the user clicks any item in the statusbar. <u>Syntax:</u> <i>referencevariable.ItemClicked += methodname;</i>

Properties of "ToolStripStatusLabel" class:

The "ToolStripStatusLabel" class's object represents a label in the statusbar.

Sl. No	Property	Description
1	Text	Represents text of the toolStrip status label. <u>Syntax:</u> <i>referencevariable.Text = "string here";</i>

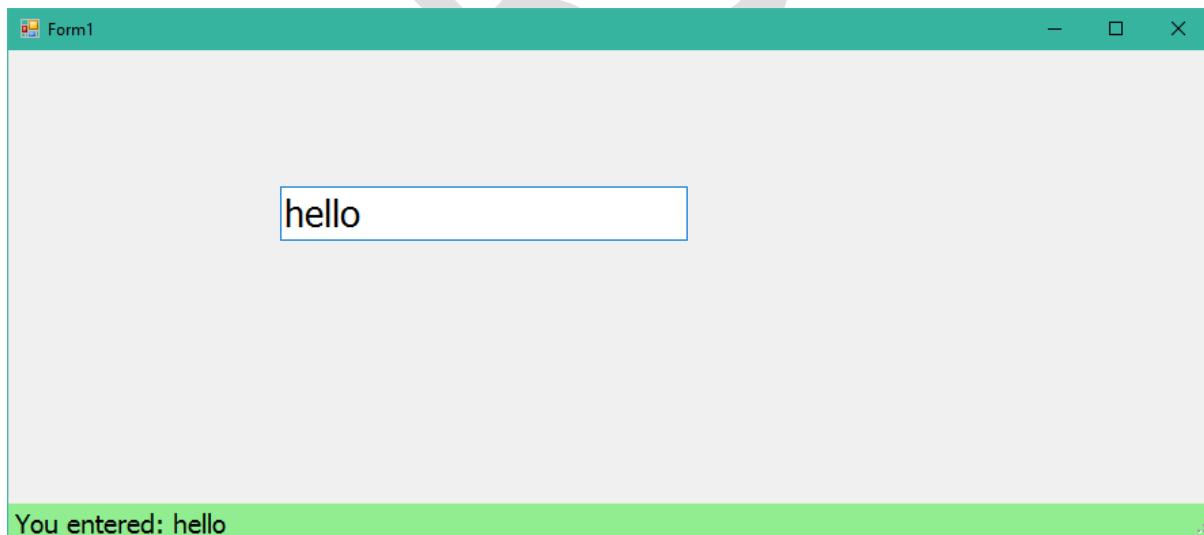
2	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>
4	Image	Represents icon image of the toolbar status label. <u>Syntax:</u> <code>referencevariable.Image = System.Drawing.Image.FromFile("image file path here");</code>
5	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
6	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>
7	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>
8	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <code>referencevariable.Enabled = true false;</code>

Events of "ToolStripStatusLabel" class:

Sl. No	Event	Description
1	Click	Executes when the user clicks on the toolbar status label. <u>Syntax:</u> <code>referencevariable.Click += methodname;</code>
2	MouseEnter	Executes when the user places the mouse pointer on the toolbar status label. <u>Syntax:</u> <code>referencevariable.MouseEnter += methodname;</code>
3	MouseLeave	Executes when the user moves the mouse pointer out of the toolbar status label. <u>Syntax:</u> <code>referencevariable.MouseLeave += methodname;</code>

“System.Windows.Forms.StatusStrip” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “StatusStripExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StatusStripExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace StatusStripExample
{
    public partial class Form1 : Form
    {
        TextBox textbox1;
        StatusStrip statusstrip1;
        ToolStripStatusLabel toolstripstatuslabel1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(900, 400);

            /* textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(300, 40);
            textbox1.Location = new Point(200, 100);
```

```
textbox1.Text = "Your text here";
textbox1.TextChanged += Textbox1_TextChanged;
this.Controls.Add(textBox1);

/* statusstrip1 */
statusstrip1 = new StatusStrip();
statusstrip1.Dock = DockStyle.Bottom;
statusstrip1.BackColor = Color.LightGreen;
statusstrip1.Font = new Font("Tahoma", 14);
this.Controls.Add(statusstrip1);

/* toolstripstatuslabel1 */
toolStripStatusLabel1 = new ToolStripStatusLabel();
toolStripStatusLabel1.Text = "Ready";
statusstrip1.Items.Add(toolstripStatusLabel1);

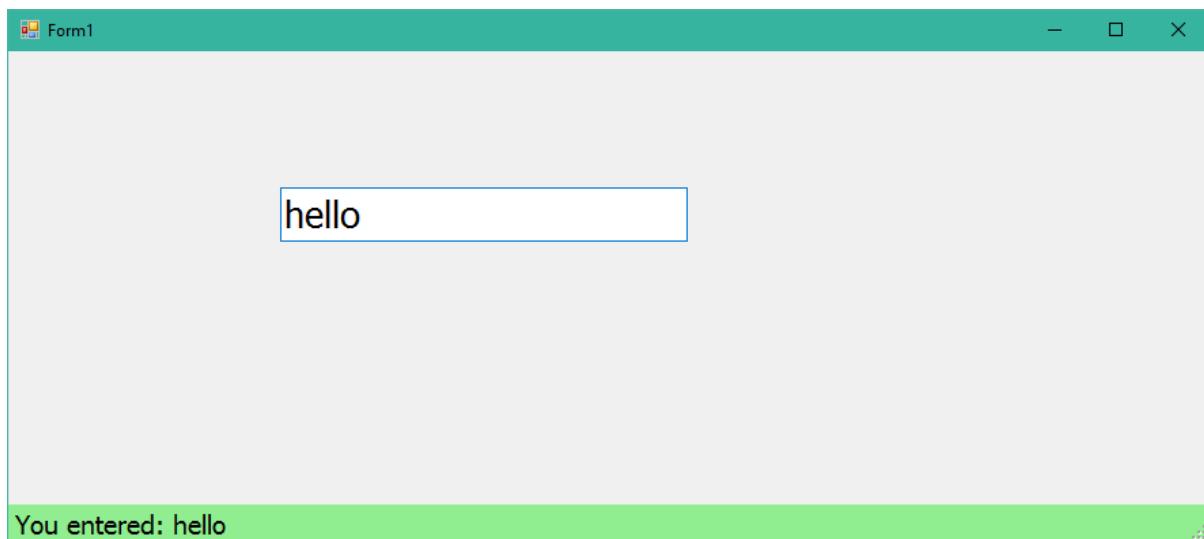
}

private void Textbox1_TextChanged(object sender, EventArgs e)
{
    toolStripStatusLabel1.Text = "You entered: " + textBox1.Text;
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Type something in the textbox, it shows the same in the status bar.

“System.Windows.Forms.RichTextBox” class

The “System.Windows.Forms.RichTextBox” class

- “RichTextBox” is a class; “System.Windows.Forms” is a namespace.
- The “RichTextBox” class / control is used to create a text editor application such as “Wordpad”.
- RichTextBox supports “rtf” (Rich Text Format).

Steps for development of RichTextBox:

- Import the namespace:
 - `using System.Windows.Forms;`
- Create a reference variable:
 - `RichTextBox referencevariable;`

- Create an object:
 - `referencevariable = new RichTextBox();`

- Set properties:
 - `referencevariable.property = value;`

- Add event:
 - `referencevariable.event += method;`
- Add control to the form:
 - `this.Controls.Add(referencevariable);`

Properties of "RichTextBox" class:

Sl. No	Property	Description
1	Text	Represents text of the control. <u>Syntax:</u> <code>referencevariable.Text = "text here";</code>
2	Font	Represents font settings of the control. <u>Syntax:</u> <code>referencevariable.Font = new System.Drawing.Font(string FontName, int FontSize);</code>
3	Size	Represents size (width and height) of the control. <u>Options:</u> pixels. <u>Syntax:</u> <code>referencevariable.Size = new System.Drawing.Size(int width, int height);</code>

4	MaxLength	Represents the maximum number of characters that can be entered in the control. <u>Syntax:</u> <code>referencevariable.MaxLength = int number;</code>
5	Location	Represents position (X and Y) of the control on the form. <u>Options:</u> pixels <u>Syntax:</u> <code>referencevariable.Location = new System.Drawing.Point(int x, int y);</code>
6	Cursor	Represents mouse cursor of the control. <u>Options:</u> Arrow Hand etc. <u>Syntax:</u> <code>referencevariable.Cursor = System.Windows.Forms.Cursors.Hand;</code>
7	BackColor	Represents background color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.BackColor = System.Drawing.Color.Green;</code>
8	ForeColor	Represents foreground color of the control. <u>Options:</u> Green Red Blue etc. <u>Syntax:</u> <code>referencevariable.ForeColor = System.Drawing.Color.Green;</code>
9	Visible	<u>True:</u> The control appears on the form. <u>False:</u> The control will disappear on the form. <u>Syntax:</u> <code>referencevariable.Visible = true false;</code>
10	Enabled	<u>True:</u> The control works. <u>False:</u> The control doesn't work. That means, the control can't be used by the user. <u>Syntax:</u> <code>referencevariable.Enabled = true false;</code>

11	Readonly	<p><u>True:</u> The user can't modify the value of textbox.</p> <p><u>False:</u> The user can modify the value of textbox.</p> <p><u>Syntax:</u> <code>referencevariable.Readonly = true false;</code></p>
12	TabIndex	<p>Represents tab order. Ex: 1, 2, 3 etc.</p> <p><u>Syntax:</u> <code>referencevariable.TabIndex = any number;</code></p>
13	Dock	<p>Displays the control at most top / left / right / bottom side of the form.</p> <p><u>Syntax:</u> <code>referencevariable.Dock = System.Windows.Forms.DockStyle.Optionhere;</code></p>
14	SelectionFont	<p>Represents font settings of the selected text in the control.</p> <p><u>Syntax:</u> <code>referencevariable.SelectionFont = new System.Drawing.Font(string FontName, int FontSize);</code></p>
15	SelectedText	<p>Represents the selected text of the control.</p> <p><u>Syntax:</u> <code>referencevariable.SelectedText = "text here";</code></p>
16	SelectionBackColor	<p>Represents background color of the selected text of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.SelectionBackColor = System.Drawing.Color.Green;</code></p>
17	SelectionColor	<p>Represents foreground color of the selected text of the control.</p> <p><u>Options:</u> Green Red Blue etc.</p> <p><u>Syntax:</u> <code>referencevariable.SelectionColor = System.Drawing.Color.Green;</code></p>

Events of “RichTextBox” class:

Sl. No	Event	Description

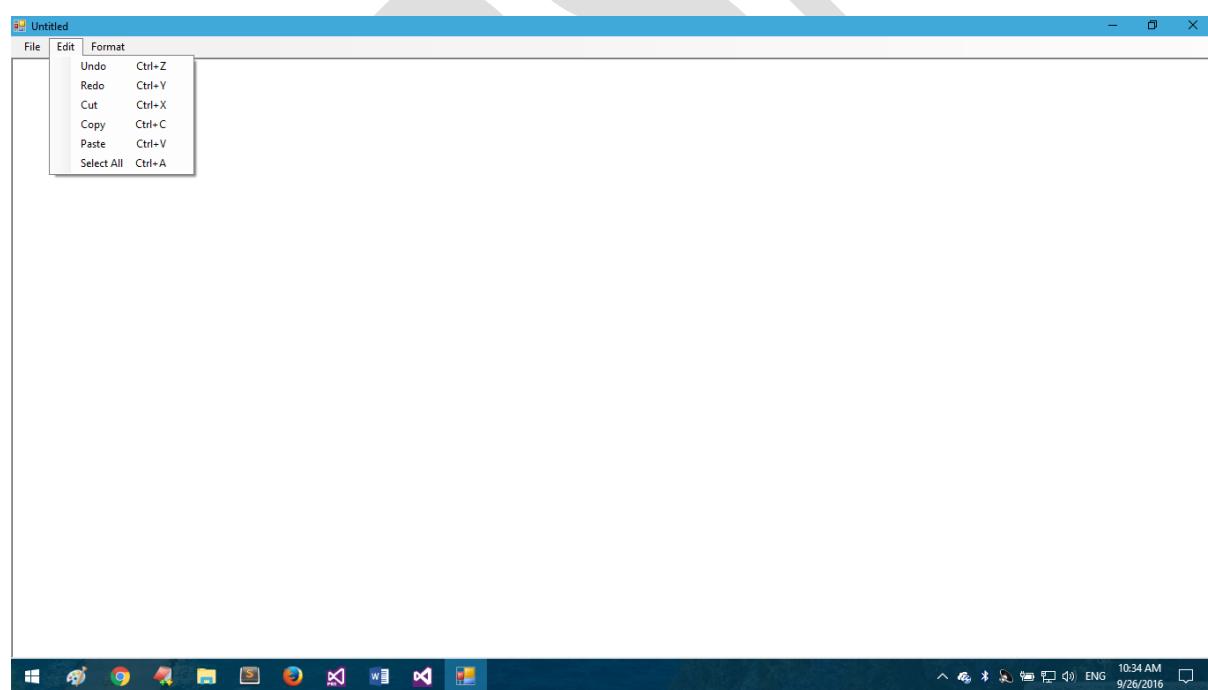
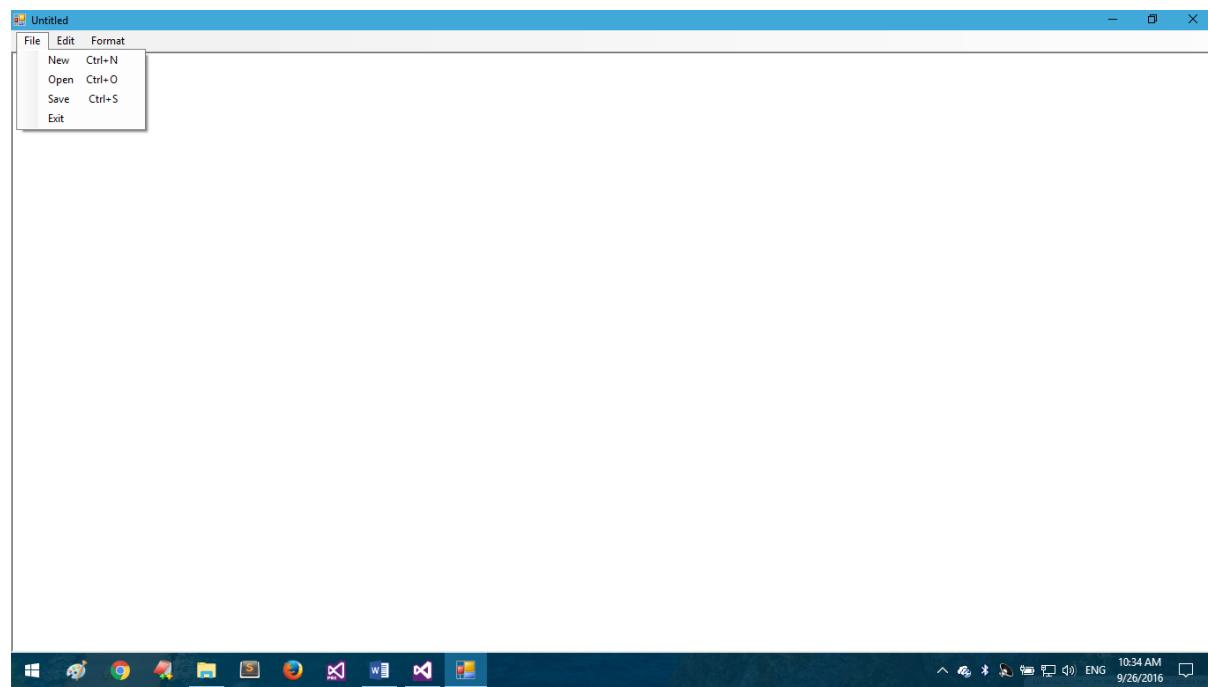
1	TextChanged	Executes when the user types a character in the control. <u>Syntax:</u> <i>referencevariable.TextChanged += methodname;</i>
2	Enter	Executes when the cursor enters into the control. <u>Syntax:</u> <i>referencevariable.Enter += methodname;</i>
3	Leave	Executes when the cursor leaves into the control. <u>Syntax:</u> <i>referencevariable.Leave += methodname;</i>
4	Click	Executes when the user clicks on the control. <u>Syntax:</u> <i>referencevariable.Click += methodname;</i>
5	KeyPress	Executes when the user presses any key on the keyboard (before accepting the character into the textbox). <u>Syntax:</u> <i>referencevariable.KeyPress += methodname;</i>

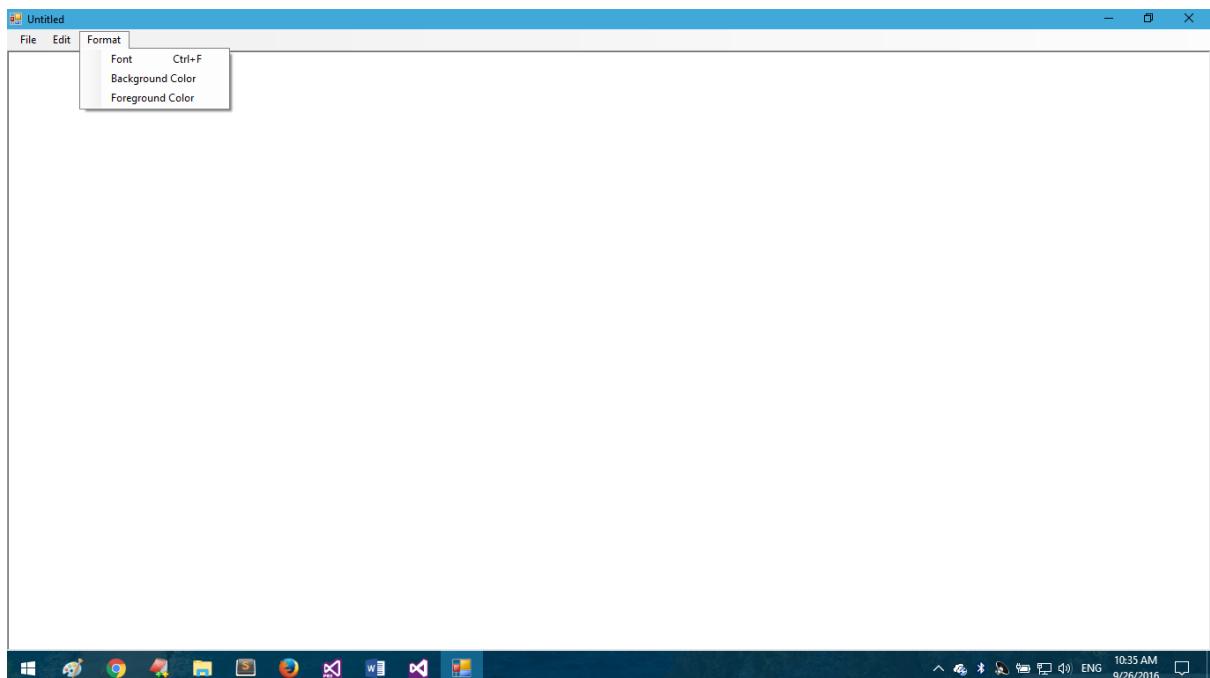
Methods of "RichTextBox" class:

Sl. No	Method	Description
1	Clear()	Clears all text in the textbox. <u>Syntax:</u> <code>referencevariable.Clear();</code>
2	SelectAll()	Selects all text in the textbox. <u>Syntax:</u> <code>referencevariable.SelectAll();</code>
3	Cut()	Cuts the selected text in the textbox. <u>Syntax:</u> <code>referencevariable.Cut();</code>
4	Copy()	Copies the selected text in the textbox. <u>Syntax:</u> <code>referencevariable.Copy();</code>
5	Paste()	Pastes the cut / copied text in the textbox. <u>Syntax:</u> <code>referencevariable.Paste()</code>
6	Undo()	Undos the previous action in the textbox. <u>Syntax:</u> <code>referencevariable.Undo()</code>
7	Redo()	Redos the previous undo action in the textbox. <u>Syntax:</u> <code>referencevariable.Undo()</code>
8	LoadFile()	Opens an existing rtf file and displays the text in the richtextbox. <u>Syntax:</u> <code>referencevariable.LoadFile("file path")</code>
9	SaveFile()	Saves the text of the richtextbox into an rtf file. <u>Syntax:</u> <code>referencevariable.SaveFile("file path")</code>

“System.Windows.Forms. RichTextBox” class - Example

Expected Output





Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “RichTextBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RichTextBoxExample”.
- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

using System;

```

using System.Drawing;
using System.Windows.Forms;

namespace RichTextBoxExample
{
    public partial class Form1 : Form
    {
        string filename = "Untitled";

        RichTextBox richtextbox1;
        MenuStrip menustrip1;
        ToolStripMenuItem filetoolStripmenuItem, newtoolStripmenuItem,
        opentoolStripmenuItem, savetoolStripmenuItem, exittoolStripmenuItem,
        edittoolStripmenuItem, undotoolStripmenuItem, redotoolStripmenuItem,
        cuttoolStripmenuItem, copytoolStripmenuItem, pastetoolStripmenuItem,
        selectalltoolStripmenuItem, formattoolStripmenuItem,
        fonttoolStripmenuItem, backgroundcolortoolStripmenuItem,
        foregroundcolortoolStripmenuItem;

        OpenFileDialog openfiledialog1;
        SaveFileDialog savefiledialog1;
        FontDialog fontdialog1;
        ColorDialog colordialog1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.WindowState = FormWindowState.Maximized;
            this.Text = filename;
            this.FormClosing += Form1_FormClosing;

            /* richtextbox */
            richtextbox1 = new RichTextBox();
            richtextbox1.Dock = DockStyle.Fill;
            richtextbox1.TextChanged += Richtextbox1_TextChanged;
    }
}

```

```

this.Controls.Add(richtextbox1);

/* menustrip1 */
menustrip1 = new MenuStrip();
this.Controls.Add(menustrip1);

/* filetoolstripmenuItem */
filetoolstripmenuItem = new ToolStripMenuItem();
filetoolstripmenuItem.Text = "&File";
menustrip1.Items.Add(filetoolstripmenuItem);

/* edittoolstripmenuItem */
edittoolstripmenuItem = new ToolStripMenuItem();
edittoolstripmenuItem.Text = "&Edit";
menustrip1.Items.Add(edittoolstripmenuItem);

/* formattoolStripmenuItem */
formattoolStripmenuItem = new ToolStripMenuItem();
formattoolStripmenuItem.Text = "F&ormat";
menustrip1.Items.Add(formattoolStripmenuItem);

/* newtoolStripmenuItem */
newtoolStripmenuItem = new ToolStripMenuItem();
newtoolStripmenuItem.Text = "&New";
newtoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | Keys.N);
newtoolStripmenuItem.Click += NewtoolStripmenuItem_Click;
filetoolstripmenuItem.DropDownItems.Add(newtoolStripmenuItem);

/* opentoolStripmenuItem */
opentoolStripmenuItem = new ToolStripMenuItem();
opentoolStripmenuItem.Text = "&Open";
opentoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | 
Keys.O);
opentoolStripmenuItem.Click += OpentoolStripmenuItem_Click;

filetoolstripmenuItem.DropDownItems.Add(opentoolStripmenuItem);

```

```

/* savetoolstripmenuItem */
savetoolStripmenuItem = new ToolStripMenuItem();
savetoolStripmenuItem.Text = "&Save";
savetoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | Keys.S);
savetoolStripmenuItem.Click += SavetoolStripmenuItem_Click;
filetoolStripmenuItem.DropDownItems.Add(savetoolStripmenuItem);

/* exittoolStripmenuItem */
exittoolStripmenuItem = new ToolStripMenuItem();
exittoolStripmenuItem.Text = "E&xit";
exittoolStripmenuItem.Click += ExittoolStripmenuItem_Click;
filetoolStripmenuItem.DropDownItems.Add(exittoolStripmenuItem);

/* undotoolStripmenuItem */
undotoolStripmenuItem = new ToolStripMenuItem();
undotoolStripmenuItem.Text = "&Undo";
undotoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | 
Keys.Z);
undotoolStripmenuItem.Click += UndotoolStripmenuItem_Click;

edittoolStripmenuItem.DropDownItems.Add(undotoolStripmenuItem);

/* redotoolStripmenuItem */
redotoolStripmenuItem = new ToolStripMenuItem();
redotoolStripmenuItem.Text = "&Redo";
redotoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | Keys.Y);
redotoolStripmenuItem.Click += RedotoolStripmenuItem_Click;

edittoolStripmenuItem.DropDownItems.Add(redotoolStripmenuItem);

/* cuttoolStripmenuItem */
cuttoolStripmenuItem = new ToolStripMenuItem();
cuttoolStripmenuItem.Text = "Cu&t";
cuttoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | Keys.X);
cuttoolStripmenuItem.Click += CuttoolStripmenuItem_Click;
edittoolStripmenuItem.DropDownItems.Add(cuttoolStripmenuItem);

```

```

/* copytoolStripmenuItem */
copytoolStripmenuItem = new ToolStripMenuItem();
copytoolStripmenuItem.Text = "&Copy";
copytoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | Keys.C);
copytoolStripmenuItem.Click += CopytoolStripmenuItem_Click;

edittoolStripMenuItem.DropDownItems.Add(copytoolStripmenuItem);

/* pastetoolStripmenuItem */
pastetoolStripmenuItem = new ToolStripMenuItem();
pastetoolStripmenuItem.Text = "&Paste";
pastetoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | 
Keys.V);
pastetoolStripmenuItem.Click += PastetoolStripmenuItem_Click;

edittoolStripMenuItem.DropDownItems.Add(pastetoolStripmenuItem);

/* selectalltoolStripmenuItem */
selectalltoolStripmenuItem = new ToolStripMenuItem();
selectalltoolStripmenuItem.Text = "Select &All";
selectalltoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | 
Keys.A);
selectalltoolStripmenuItem.Click += SelectalltoolStripmenuItem_Click;

edittoolStripMenuItem.DropDownItems.Add(selectalltoolStripmenuItem);

/* fonttoolStripmenuItem */
fonttoolStripmenuItem = new ToolStripMenuItem();
fonttoolStripmenuItem.Text = "&Font";
fonttoolStripmenuItem.ShortcutKeys = (Keys)(Keys.Control | Keys.F);
fonttoolStripmenuItem.Click += FonttoolStripmenuItem_Click;

formattoolStripMenuItem.DropDownItems.Add(fonttoolStripmenuItem);

/* backgroundcolortoolStripmenuItem */
backgroundcolortoolStripmenuItem = new ToolStripMenuItem();
backgroundcolortoolStripmenuItem.Text = "&Background Color";

```

```

backgroundcolortoolStripMenuItem.Click +=  

BackgroundcolortoolStripMenuItem_Click;  
  

formattoolStripMenuItem.DropDownItems.Add(backgroundcolortoolStrip  

menuItem);  
  

/* foregroundcolortoolStripMenuItem */  

foregroundcolortoolStripMenuItem = new ToolStripMenuItem();  

foregroundcolortoolStripMenuItem.Text = "&Foreground Color";  

foregroundcolortoolStripMenuItem.Click +=  

ForegroundcolortoolStripMenuItem_Click;  
  

formattoolStripMenuItem.DropDownItems.Add(foregroundcolortoolStrip  

menuItem);  
  

/* openfiledialog1 */  

openfiledialog1 = new OpenFileDialog();  

openfiledialog1.Filter = "Rich Text Files|*.rtf|Text Files|*.txt";  

openfiledialog1.FilterIndex = 0;  
  

/* savefiledialog1 */  

savefiledialog1 = new SaveFileDialog();  

savefiledialog1.Filter = "Rich Text Files|*.rtf|Text Files|*.txt";  

savefiledialog1.FilterIndex = 0;  
  

/* fontdialog1 */  

fontdialog1 = new FontDialog();  
  

/* colordialog1 */  

colordialog1 = new ColorDialog();
}  
  

private void Form1_FormClosing(object sender, FormClosingEventArgs  

e)
{
    if (this.Text.EndsWith("*") == true)
    {

```

```
 DialogResult dr = MessageBox.Show("Do you want to save changes?", "RichTextBox Example", MessageBoxButtons.YesNoCancel);
if (dr == DialogResult.Yes)
{
    save();
}
else if (dr == DialogResult.No)
{
    //empty
}
else if (dr == DialogResult.Cancel)
{
    e.Cancel = true;
}
}

private void RichTextbox1_TextChanged(object sender, EventArgs e)
{
    this.Text = filename + " *";
}

private void save()
{
    if (filename == "Untitled")
    {
        DialogResult dr = saveFileDialog1.ShowDialog();
        if (dr == DialogResult.OK)
        {
            richTextBox1.SaveFile(saveFileDialog1.FileName);
        }
    }
    else
    {
        richTextBox1.SaveFile(filename);
        this.Text = filename;
    }
}
```

```
}

private void NewtoolstripmenuItem_Click(object sender, EventArgs e)
{
    if (this.Text.EndsWith("*)") == true)
    {
        DialogResult dr = MessageBox.Show("Do you want to save
changes?", "RichTextBox Example", MessageBoxButtons.YesNoCancel);
        if (dr == DialogResult.Yes)
        {
            save();

            richtextbox1.Clear();
            filename = "Untitled";
            this.Text = filename;
        }
        else if (dr == DialogResult.No)
        {
            richtextbox1.Clear();
            filename = "Untitled";
            this.Text = filename;
        }
        else if (dr == DialogResult.Cancel)
        {
            //empty
        }
    }
    else
    {
        richtextbox1.Clear();
        filename = "Untitled";
        this.Text = filename;
    }
}
private void OpentoolstripmenuItem_Click(object sender, EventArgs e)
{
    if (this.Text.EndsWith("*)") == true)
```

```
{  
    DialogResult dr = MessageBox.Show("Do you want to save  
changes?", "RichTextBox Example", MessageBoxButtons.YesNoCancel);  
    if (dr == DialogResult.Yes)  
    {  
        save();  
  
        DialogResult dr2 = openfiledialog1.ShowDialog();  
        if (dr2 == DialogResult.OK)  
        {  
            richtextbox1.LoadFile(openfiledialog1.FileName);  
            filename = openfiledialog1.FileName;  
            this.Text = filename;  
        }  
    }  
    else if (dr == DialogResult.No)  
    {  
        DialogResult dr2 = openfiledialog1.ShowDialog();  
        if (dr2 == DialogResult.OK)  
        {  
            richtextbox1.LoadFile(openfiledialog1.FileName);  
            filename = openfiledialog1.FileName;  
            this.Text = filename;  
        }  
    }  
    else if (dr == DialogResult.Cancel)  
    {  
        //empty  
    }  
}  
else  
{  
    DialogResult dr2 = openfiledialog1.ShowDialog();  
    if (dr2 == DialogResult.OK)  
    {  
        richtextbox1.LoadFile(openfiledialog1.FileName);  
        filename = openfiledialog1.FileName;  
    }  
}
```

```
        this.Text = filename;
    }
}
}
private void SavetoolstripmenuItem_Click(object sender, EventArgs e)
{
    save();
}

private void ExittoolStripmenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}
private void UndotoolStripmenuItem_Click(object sender, EventArgs e)
{
    richtextbox1.Undo();
    this.Text = filename + " *";
}
private void RedotoolStripmenuItem_Click(object sender, EventArgs e)
{
    richtextbox1.Redo();
    this.Text = filename + " *";
}
private void CuttoolStripmenuItem_Click(object sender, EventArgs e)
{
    richtextbox1.Cut();
    this.Text = filename + " *";
}
private void CopytoolStripmenuItem_Click(object sender, EventArgs e)
{
    richtextbox1.Copy();
    this.Text = filename + " *";
}
private void PastetoolStripmenuItem_Click(object sender, EventArgs e)
{
    richtextbox1.Paste();
    this.Text = filename + " *";
}
```

```
}

private void SelectalltoolStripmenuItem_Click(object sender,
EventArgs e)
{
    richtextbox1.SelectAll();
}

private void FonttoolStripmenuItem_Click(object sender, EventArgs e)
{
    fontdialog1.Font = richtextbox1.SelectionFont;
    if (fontdialog1.ShowDialog() == DialogResult.OK)
    {
        richtextbox1.SelectionFont = fontdialog1.Font;
        this.Text = filename + " *";
    }
}

private void BackgroundcolortoolStripmenuItem_Click(object sender,
EventArgs e)
{
    colordialog1.Color = richtextbox1.SelectionBackColor;
    if (colordialog1.ShowDialog() == DialogResult.OK)
    {
        richtextbox1.SelectionBackColor = colordialog1.Color;
        this.Text = filename + " *";
    }
}

private void ForegroundcolortoolStripmenuItem_Click(object sender,
EventArgs e)
{
    colordialog1.Color = richtextbox1.SelectionColor;
    if (colordialog1.ShowDialog() == DialogResult.OK)
    {
        richtextbox1.SelectionColor = colordialog1.Color;
        this.Text = filename + " *";
    }
}
```

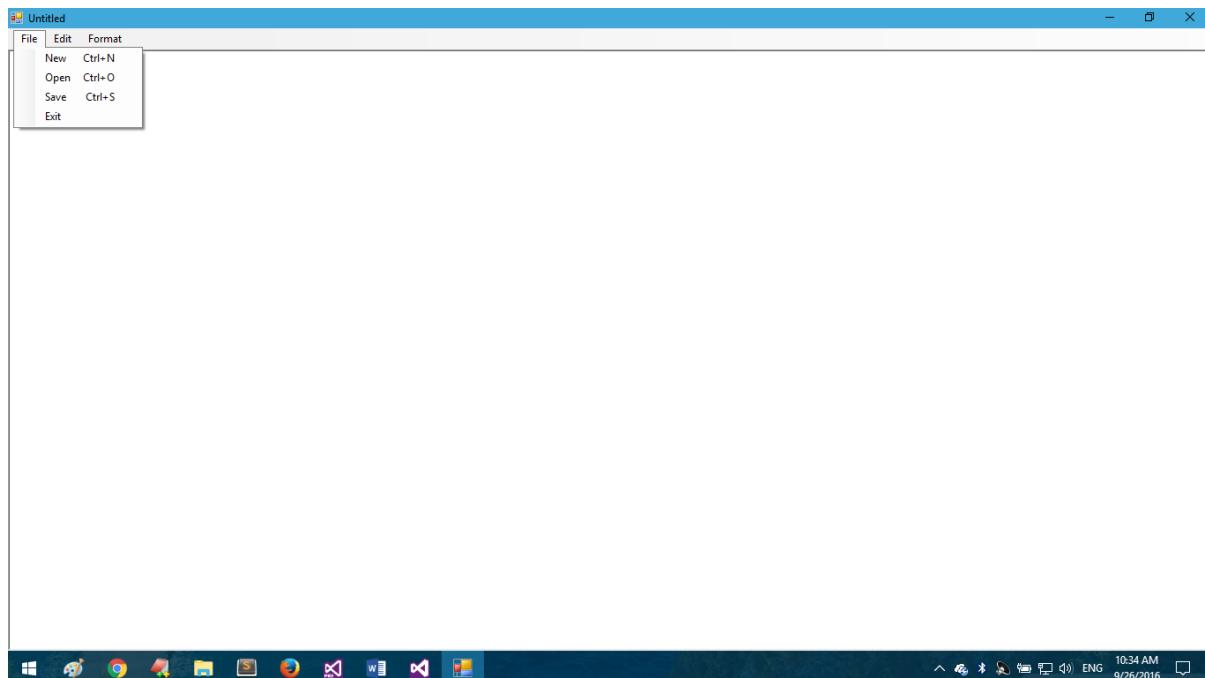
```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Try to click on all the options in “File”, “Edit”, “Format” menus.

User Controls

UserControls

- .NET provides a set of pre-defined controls, which are enough for most of the regular application development. In large-scale applications, we may want to customize the existing controls. Then we will create child classes for existing control classes such as Label, TextBox, ComboBox.
- The UserControls are child classes that are created based on existing control classes such as Label, TextBox, ComboBox etc.

- In the user controls, we can define additional properties, methods, events and also override existing properties and methods.

Steps for development of UserControl:

- Create a user control class:

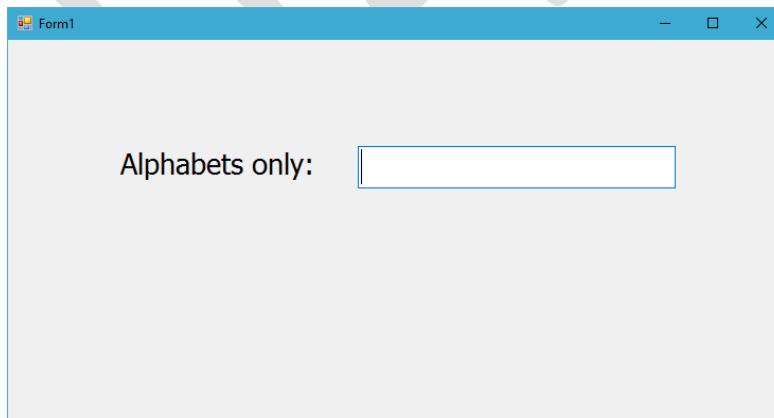
```
class childclassname : controlclassname
{
    public childclassname()
    {
        InitializeComponent();
    }
    Additional properties, methods and events
    Override existing properties, methods
}
```

- Create an object for the user control class:

```
childclassname referencevariable = new childclassname();
```

User Controls - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to "File" – "New" – "Project".

- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “UserControlsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UserControlsExample”.
- Click on OK.
- Right click on the form and click on “View Code”.
- Open solution explorer. Right click on the project name (UserControlsExample) and click on “Add” – “New Item” – “Windows Forms” – “User Control”. Type the file name as “AlphabeticTextBox.cs”. Click on “Add”. Right click on the user control and click on “View Code”.

AlphabeticTextBox.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace UserControlsExample
{
    public partial class AlphabeticTextBox : TextBox
    {
        public AlphabeticTextBox()
        {
            InitializeComponent();
            this.KeyPress += AlphabeticTextBox_KeyPress;
        }

        private void AlphabeticTextBox_KeyPress(object sender,
        KeyPressEventArgs e)
    }
```

```

    if ((e.KeyChar >= 65 && e.KeyChar <= 90) || (e.KeyChar >= 97 &&
e.KeyChar <= 122) || (e.KeyChar == 32) || (e.KeyChar == 8) || (e.KeyChar == 0))
        e.Handled = false;
    else
        e.Handled = true;
}
}
}

```

AlphabeticTextBox.Designer.cs

- Open solution explorer. Expand “AlphabeticTextBox.cs” and double click on “AlphabeticTextBox.Designer.cs”.

```

namespace UserControlsExample
{
    partial class AlphabeticTextBox
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}

```

```
#region Component Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    components = new System.ComponentModel.Container();
    //this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
}

#endregion
}
}
```

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace UserControlsExample
{
    public partial class Form1 : Form
    {
        Label label1;
        AlphabeticTextBox alphabetictextbox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(750, 400);
        }
    }
}
```

```
/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Location = new Point(100, 100);
label1.Text = "Alphabets only:";
this.Controls.Add(label1);

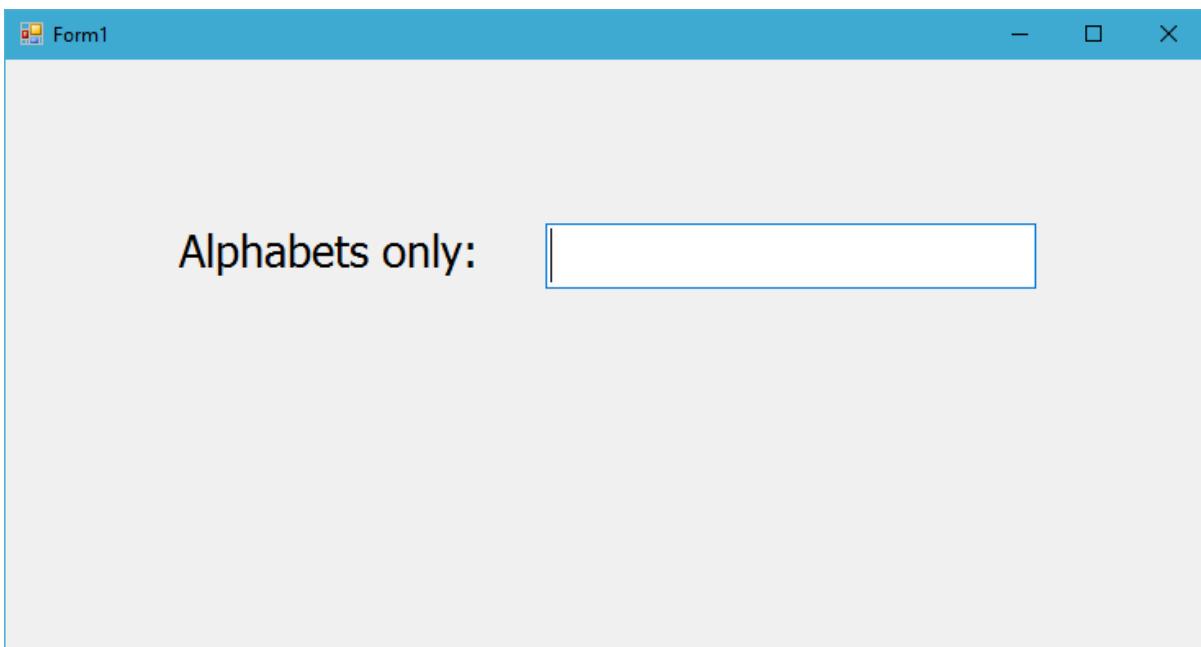
/* alphabetictextbox1 */
alphabetictextbox1 = new AlphabeticTextBox();
alphabetictextbox1.Location = new Point(330, 100);
alphabetictextbox1.Size = new Size(300, 100);
this.Controls.Add(alphabetictextbox1);

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



You can type only alphabets in the textbox.

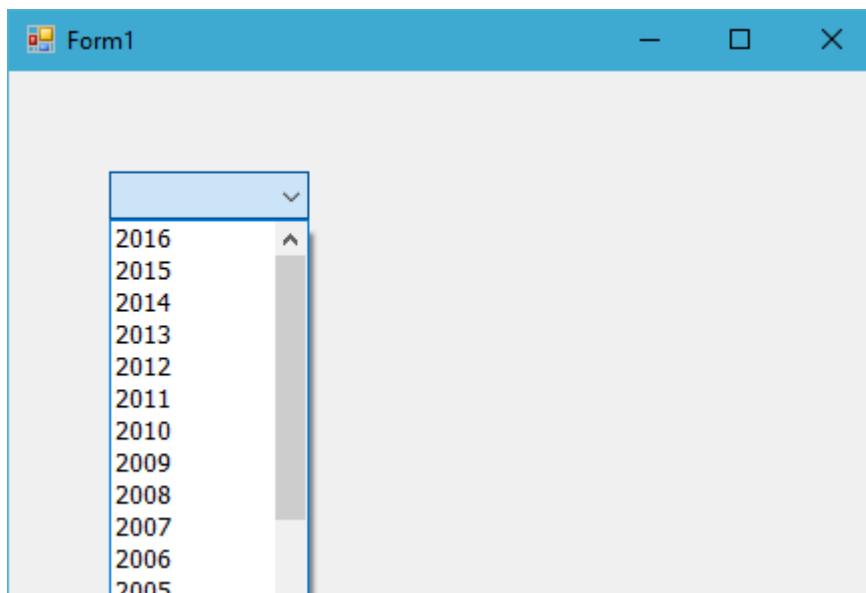
Windows Forms Control Library

Windows Forms Control Library

- “Windows Forms Control Library” is a collection of user controls, which can be used in many windows forms applications.
- Windows Forms Control Library is a “project template” in visual studio.
- Windows Forms Control Library will be compiled into a dll file. We can add the reference of the dll (Dynamic Link Library) file into the windows forms applications & call all the user controls that are created in the windows forms control library.

Windows Forms Control Library - Example

Expected Output



Creating Windows Forms Control Library

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Windows” – “Classic Desktop”.
- Select “Windows Forms Control Library”.
- Type the project name as “WindowsFormsControlLibrary1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WindowsFormsControlLibrary1”.
- Click on OK.
- Right click on the form and click on “View Code”.
- Open solution explorer. Right click on the user control file (UserControl1.cs) and click on “Rename”. Type the file name as “YearComboBox.cs” and press Enter. Right click on the user control and click on “View Code”.

YearComboBox.cs

```
using System;
using System.Drawing;
```

```

using System.Collections.Generic;
using System.Windows.Forms;

namespace WindowsFormsControlLibrary1
{
    public partial class YearComboBox : ComboBox
    {
        public YearComboBox()
        {
            InitializeComponent();

            List<object> years = new List<object>();
            for (int i = DateTime.Now.Year; i >= DateTime.Now.Year - 100; i--)
            {
                years.Add(i);
            }

            this.Items.AddRange(years.ToArray());
        }
    }
}

```

YearComboBox.Designer.cs

- Open solution explorer. Expand “YearComboBox.cs” and double click on “YearComboBox.Designer.cs”.

```

namespace WindowsFormsControlLibrary1
{
    partial class YearComboBox
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

```

```

/// <summary>
/// Clean up any resources being used.
/// </summary>
/// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
protected override void Dispose(bool disposing)
{
    if(disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#region Component Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    components = new System.ComponentModel.Container();
    //this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
}

#endregion
}
}

```

Compiling the windows forms control library:

- Go to “Build” menu – “Build solution”.
- It generates the dll file at the following location:

C:\CSharp\WindowsFormsControlLibrary1\WindowsFormsControlLibrary1\bin\
 Debug\WindowsFormsControlLibrary1.dll

Creating Windows Forms Application

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “WindowsFormsApplication1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WindowsFormsApplication1”.
- Click on OK.
- Right click on the form and click on “View Code”.
- Open solution explorer. Right click on the project name (WindowsFormsApplication1) and click on “Add” – “Reference”. Click on “Browse”. Select the dll file “C:\CSharp\WindowsFormsControlLibrary1\WindowsFormsControlLibrary1\bin\Debug\WindowsFormsControlLibrary1.dll”. Click on “Add”. Click on “OK”. It adds “WindowsFormsControlLibrary1” in the “References” list in the solution explorer.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using WindowsFormsControlLibrary1;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        YearComboBox yearcombobox;

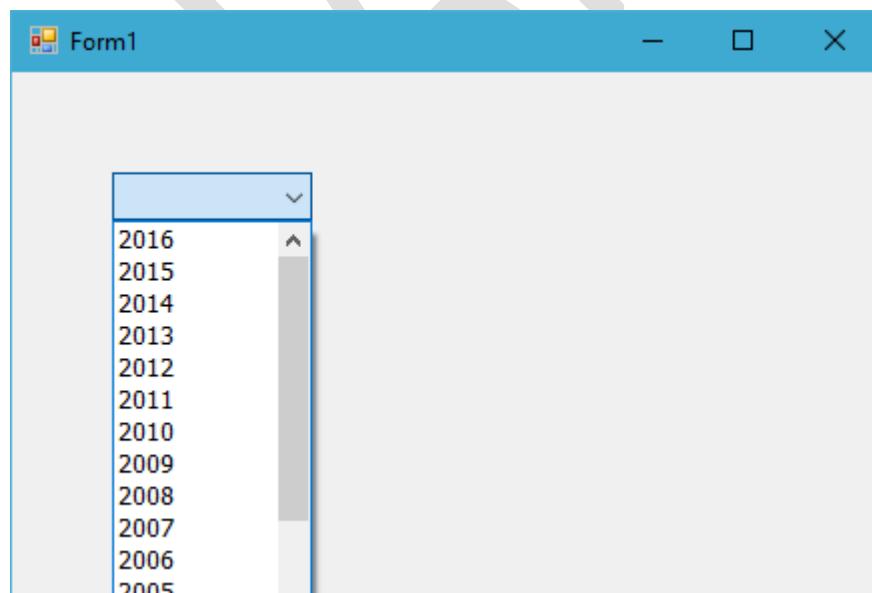
        public Form1()
    }
}
```

```
{  
    InitializeComponent();  
  
    this.Font = new Font("Tahoma", 10);  
    this.Size = new Size(450, 300);  
  
    /* yearcombobox1 */  
    yearcombobox = new YearComboBox();  
    yearcombobox.DropDownStyle = ComboBoxStyle.DropDownList;  
    yearcombobox.Location = new Point(50, 50);  
    yearcombobox.Size = new Size(100, 50);  
  
    this.Controls.Add(yearcombobox);  
}  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Years are automatically added in the combobox.

“System.Net.Mail.SmtpClient” class

The “System.Net.Mail.SmtpClient” class

- The “SmtpClient” is a class, which is a member of “System.Net.Mail” namespace.
- This class is used to send an email to any mail.

Properties of “SmtpClient” class:

Sl. No	Property	Description
1	EnableSsl	Enables “Security Socket Layer (SSL)”, that means “HTTPS” (HTTP with Security). <u>Syntax:</u> <i>referencevariable.Ssl = true;</i>
2	Host	Represents mail server name. <u>Syntax:</u> <i>referencevariable.Host = “host name”;</i>
3	Port	Represents mail server port number. <u>Syntax:</u> <i>referencevariable.Port = any port;</i>
4	Credentials	Represents an object of “NetworkCredential” class, which represents username and password. <u>Syntax:</u> <i>referencevariable.Credentials = new NetworkCredential(<i>username</i>, <i>password</i>);</i>

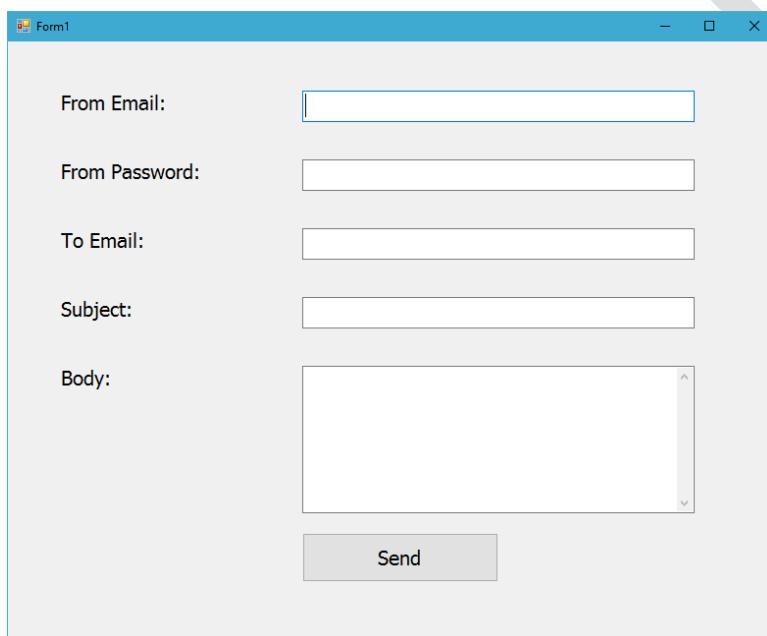
Methods of “SmtpClient” class:

Sl. No	Method	Description

1	Send <u>Syntax:</u> <i>referencevariable.Send(string from, string to, string subject, string body);</i>	Send a mail to the specified address.
---	---	---------------------------------------

“System.Net.Mail.SmtpClient” class - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SmtpExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SmtpExample”.

- Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Net.Mail;
using System.Net;

namespace SmtpExample
{
    public partial class Form1 : Form
    {
        Label labelFromEmail, labelFromPassword, labelToEmail, labelSubject,
        labelBody;
        TextBox textBoxFromEmail, textBoxFromPassword, textBoxToEmail,
        textBoxSubject, textBoxBody;
        Button buttonSend;

        public Form1()
        {
            InitializeComponent();

            this.Font = new Font("Tahoma", 15);
            this.Size = new Size(800, 650);
            this.StartPosition = FormStartPosition.CenterScreen;

            /* labelFromEmail */
            labelFromEmail = new Label();
            labelFromEmail.Text = "From Email:";
            labelFromEmail.AutoSize = true;
            labelFromEmail.Location = new Point(50, 50);
            this.Controls.Add(labelFromEmail);
```

```
/* labelFromPassword */
labelFromPassword = new Label();
labelFromPassword.Text = "From Password:";
labelFromPassword.AutoSize = true;
labelFromPassword.Location = new Point(50, 120);
this.Controls.Add(labelFromPassword);

/* labelToEmail */
labelToEmail = new Label();
labelToEmail.Text = "To Email:";
labelToEmail.AutoSize = true;
labelToEmail.Location = new Point(50, 190);
this.Controls.Add(labelToEmail);

/* labelSubject */
labelSubject = new Label();
labelSubject.Text = "Subject:";
labelSubject.AutoSize = true;
labelSubject.Location = new Point(50, 260);
this.Controls.Add(labelSubject);

/* labelBody */
labelBody = new Label();
labelBody.Text = "Body:";
labelBody.AutoSize = true;
labelBody.Location = new Point(50, 330);
this.Controls.Add(labelBody);

/* textboxFromEmail */
textboxFromEmail = new TextBox();
textboxFromEmail.Size = new Size(400, 50);
textboxFromEmail.Location = new Point(300, 50);
this.Controls.Add(textboxFromEmail);

/* textboxFromPassword */
textboxFromPassword = new TextBox();
textboxFromPassword.Size = new Size(400, 50);
```

```

textboxFromPassword.Location = new Point(300, 120);
this.Controls.Add(textboxFromPassword);

/* textboxToEmail */
textboxToEmail = new TextBox();
textboxToEmail.Size = new Size(400, 50);
textboxToEmail.Location = new Point(300, 190);
this.Controls.Add(textboxToEmail);

/* textboxSubject */
textboxSubject = new TextBox();
textboxSubject.Size = new Size(400, 50);
textboxSubject.Location = new Point(300, 260);
this.Controls.Add(textboxSubject);

/* textboxBody */
textboxBody = new TextBox();
textboxBody.Size = new Size(400, 150);
textboxBody.Location = new Point(300, 330);
textboxBody.Multiline = true;
textboxBody.ScrollBars = ScrollBars.Vertical;
this.Controls.Add(textboxBody);

/* buttonSend */
buttonSend = new Button();
buttonSend.Text = "Send";
buttonSend.Size = new Size(200, 50);
buttonSend.Location = new Point(300, 500);
buttonSend.Click += ButtonSend_Click;
this.Controls.Add(buttonSend);
}

private void ButtonSend_Click(object sender, EventArgs e)
{
    SmtpClient sc = new SmtpClient();
    sc.EnableSsl = true;
    sc.Host = "smtp.gmail.com";
}

```

```

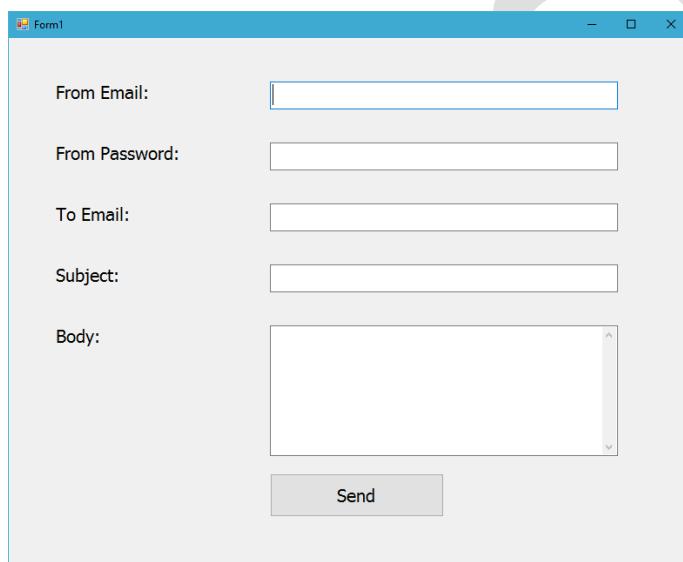
sc.Port = 587;
sc.Credentials = new NetworkCredential(textBoxFromEmail.Text,
textBoxFromPassword.Text);
sc.Send(textBoxFromEmail.Text, textBoxToEmail.Text,
textBoxSubject.Text, textBoxBody.Text);
MessageBox.Show("Sent");
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Type the details and click on “Send”.

Multi-Threading

Multi Threading

- Thread = Process = Task = Some work

- A “thread” represents a method, that can be executed in background, without effecting the main process (UI) of the application.
- That means “multi-threading” means “executing multiple threads” in background.
- Example 1: Calculating folder size in the folder properties dialog box in windows explorer.
- Example 2: Reading many records from database.

The “System.Threading.Thread” class

- “Thread” is a class, which is a member of “System.Threading” namespace.
- “Thread” class’s object represents a thread.
- “Thread” class provides a set of methods to manipulate threads at run time.

Steps for development of Multi Threading:

- Import the namespace:
 - `using System.Threading;`
- Create a reference variable:
 - `Thread referencevariable;`

- Create an object:
 - `referencevariable = new Thread(threadmethodname);`

- Set properties:
 - `referencevariable.property = value;`

- Call method:
 - `referencevariable.methodname();`

Properties of "Thread" class:

Sl. No	Property	Description
1	<code>IsBackground</code>	Specifies that the thread is a background thread or not. The background threads run in background. <u>Syntax:</u> <code>referencevariable.IsBackground = true false;</code>
2	<code>CurrentCulture</code>	Represents current culture of the current thread. This is useful for “Internationalization” only. <u>Syntax:</u> <code>referencevariable.CurrentCulture;</code>
3	<code>CurrentUICulture</code>	Represents current culture of the current thread, for resource management. This is useful for “Internationalization” only. <u>Syntax:</u> <code>referencevariable.CurrentCulture</code>

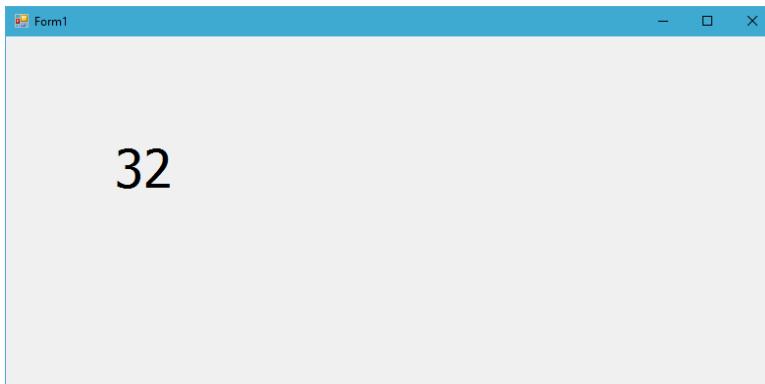
4	Priority <p>Represents priority of the current thread.</p> <p><u>Syntax:</u> Lowest BelowNormal Normal AboveNormal Highest</p> <p><u>Syntax:</u> <i>referencevariable.Priority</i> = System.Threading.ThreadPriority.OptionHere;</p>
---	--

Methods of "Thread" class:

Sl. No	Method	Description
1	Start()	Starts the thread. <u>Syntax:</u> <i>referencevariable.Start();</i>
2	Suspend()	Pauses the thread. <u>Syntax:</u> <i>referencevariable.Suspend();</i>
3	Resume()	Resumes the thread. <u>Syntax:</u> <i>referencevariable.Resume();</i>
4	Abort()	Stops the thread. <u>Syntax:</u> <i>referencevariable.Stop();</i>
5	Sleep()	Pauses the thread for specified mille seconds. <u>Syntax:</u> Thread.Sleep(int milliseconds);

Multi Threading - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MultiThreadingExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultiThreadingExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Threading;
using System.Drawing;
namespace MultiThreadingExample
{
    public partial class Form1 : Form
    {
        Thread th;
        Label lbl;
```

```
public Form1()
{
    InitializeComponent();
    /* form properties */
    this.Font = new Font("Tahoma", 30);
    this.Size = new Size(800, 400);
    this.Load += Form1_Load;
    /* label1 */
    lbl = new Label();
    lbl.Text = "1";
    lbl.AutoSize = true;
    lbl.Location = new Point(100, 100);
    lbl.Font = new Font("Tahoma", 40);
    this.Controls.Add(lbl);
}
private void Form1_Load(object sender, EventArgs e)
{
    /* multi threading */
    th = new Thread(mymethod);
    th.IsBackground = true;
    th.Start();
}

delegate void mydelegatetype();
public void mymethod()
{
    while (true)
    {
        try
        {
            if (IsDisposed == false)
            {
                this.Invoke(new mydelegatetype(mymethod2));
            }
            Thread.Sleep(50);
        }
        catch
    }
}
```

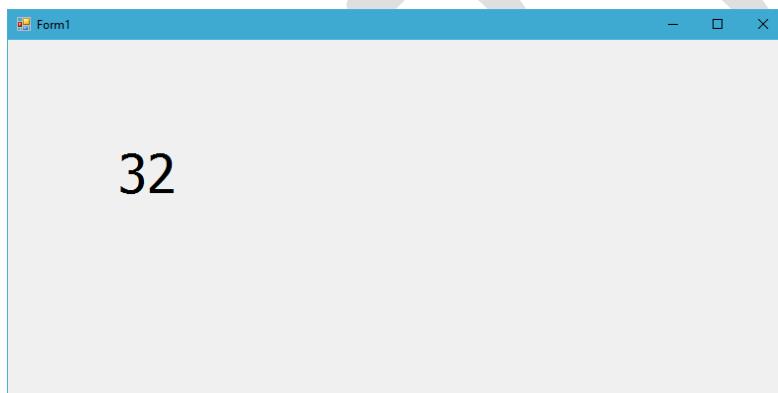
```
        {
        }
    }
}

public void mymethod2()
{
    int n = Convert.ToInt32(lbl.Text);
    n++;
    lbl.Text = Convert.ToString(n);
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It increases the counter automatically in background.

Task Parallel Library

- Task = Process = Thread = Some work
- “Task Parallel Library” is used to perform multiple processes parallelly (simultaneously) in background.

- “Task Parallel Library” is similar to “Multi Threading”; but “TPL” gives best performance in “Multi-Core” processors such as Dual Core, Core2Duo, i3, i5, i7 etc.
- Example 1: Calculating folder size in the folder properties dialog box in windows explorer.
- Example 2: Reading many records from database.

The “System.Threading.Tasks.Task” class

- “Task” is a class, which is a member of “System.Threading.Tasks” namespace.
- “Task” class’s object represents a task.
- “Task” class provides a set of methods to manipulate threads at run time.

Steps for development of Task Parallel Library:

- Import the namespace:
 - `using System.Threading.Tasks;`
- Create a reference variable:
 - `Task referencevariable;`
- Create an object:
 - `referencevariable = new Task(taskmethodname);`
- Set properties:
 - `referencevariable.property = value;`
- Call method:

- *referencevariable.methodname ()*;

Properties of “Task” class:

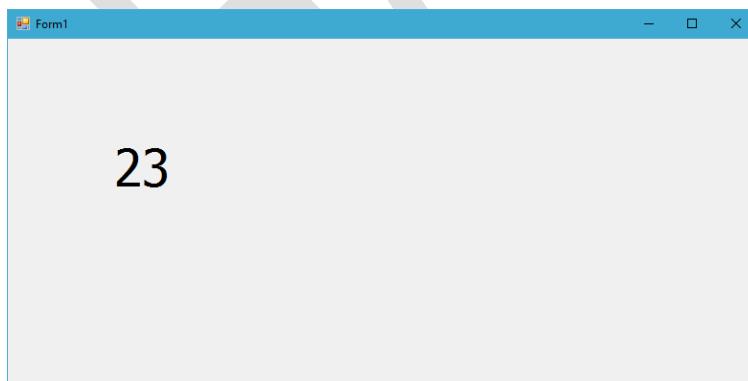
Sl. No	Property	Description
1	Status	Represents current status of the task. <u>Syntax:</u> <i>referencevariable.Task</i>

Methods of “Task” class:

Sl. No	Method	Description
1	Start()	Starts the task execution. That means it calls the method that is associated with the task. <u>Syntax:</u> <i>referencevariable.Start()</i>

Task Parallel Library - Example

Expected Output



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TpIExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TpIExample”. Click on OK.
- Right click on the form and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Threading.Tasks;
using System.Windows.Forms;

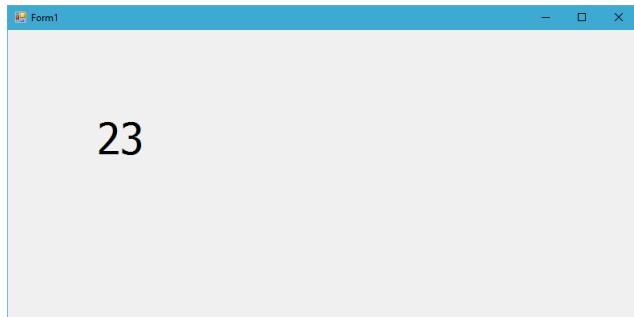
namespace TpIExample
{
    public partial class Form1 : Form
    {
        Task t;
        Label lbl;
        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 30);
            this.Size = new Size(800, 400);
            this.Load += Form1_Load;
            /* label1 */
            lbl = new Label();
            lbl.Text = "1";
            lbl.AutoSize = true;
            lbl.Location = new Point(100, 100);
            lbl.Font = new Font("Tahoma", 40);
            this.Controls.Add(lbl);
        }
    }
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    /* multi threading */
    t = new Task(mymethod);
    t.Start();
}

delegate void mydelegatetype();
public void mymethod()
{
    while (true)
    {
        try
        {
            if (IsDisposed == false)
            {
                this.Invoke(new mydelegatetype(mymethod2));
            }
            System.Threading.Thread.Sleep(50);
        }
        catch
        {
        }
    }
}
public void mymethod2()
{
    int n = Convert.ToInt32(lbl.Text);
    n++;
    lbl.Text = Convert.ToString(n);
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

It increases the counter automatically in background.

Windows Services

- Windows Services are non-gui background applications that run in background, even though there is no windows-form is running.
- Windows services are two types that “starts automatically” and “starts manually”.
 - The first type of windows services starts automatically along with windows operating system every time, we switch on the system.
 - The other type of windows services will be started manually going to Control Panel – System and Security – Administrative Tools – Services and right click on the desired windows service and click on “Start” option.
- Windows services are used to perform any continuous work such as sending e-mails for every few minutes, continuous database updates etc.
- Windows services are supported only in windows operating system.
- Examples of built-in windows services: Anti-virus programs, windows time, windows audio, plug and play, team viewer etc.

- In .net, a windows service is represented as a class that inherits from “System.ServiceProcess.ServiceBase” class.
- The “ServiceBase” class provides two virtual methods that can be implemented in the child class:
 1. OnStart(): This method executes when the user starts the windows service automatically / manually.
 2. OnStop(): This method executes when the user stops the windows service automatically / manually.

Windows Services - Example

Expected Output

An automatic email has to be sent automatically, for every 30 seconds.

Creating Windows Service Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Windows” – “Classic Desktop”.
- Select “Windows Service”.
- Type the project name as “WindowsService1”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WindowsService1”.
- Click on OK.
- It shows “Service1.cs [Design]”.
- Right click on the form and click on “View Code”.

Service1.cs

```
using System;
using System.Net;
using System.Net.Mail;
using System.ServiceProcess;
using System.Threading;

namespace WindowsService1
{
    public partial class Service1 : ServiceBase
    {
        public Service1()
        {
            InitializeComponent();
        }

        private void SendEmail()
        {
            string FromEmailID =
"harshadotnettraining1@gmail.com";
            string FromEmailPassword = "training123#";
            string ToEmailID = "harshadotnettraining2@gmail.com";
            string Subject = "Hai, this is test mail";
            string MessageBody = "The current system time is " +
DateTime.Now;

            SmtpClient client = new SmtpClient();
            client.Host = "smtp.gmail.com";
            client.Port = 587;
            client.Credentials = new NetworkCredential(FromEmailID,
FromEmailPassword);
            client.EnableSsl = true;
            client.Send(FromEmailID, ToEmailID, Subject, MessageBody); //send
mail
        }
    }
}
```

```

private void Method1()
{
    while(true)
    {
        this.SendEmail();
        Thread.Sleep(60000); //delay 60 sec (1 min)
    }
}

Thread th1;

protected override void OnStart(string[] args)
{
    th1 = new Thread(Method1);
    th1.IsBackground = true;
    th1.Start();
}

protected override void OnStop()
{
    th1.Abort();
}
}

```

Adding “ProjectInstaller.cs”

- Go to “Service1.cs [Design]”.
- Right click on the empty area and click on “Add Installer”. It automatically creates “ProjectInstaller.cs” file.
- The “ProjectInstaller.cs” file contains two components:
 - serviceInstaller1
 - serviceProcessInstaller1
- Right click on “serviceInstaller1” and click on “Properties”. Set the following properties:

- ServiceName = Automatic Emails Service
- StartType = Manual
- Right click on “serviceProcessInstaller1” and click on “Properties”. Set the following properties:
 - Account = LocalSystem

Installing the windows service

- Go to “Build” menu and click on “Build Solution”.
 - It automatically generates the “EXE” file in the following location:
C:\CSharp\WindowsService1\WindowsService1\bin\Debug\WindowsService1.exe
- Go to “Start” – “All Programs” – “Visual Studio 2015” – Right click on “Developer Command Prompt for VS 2015” and click on “Run as administrator”. Click on “Yes”.
- Type the following command in the console window:

```
installutil "C:\CSharp\WindowsService1\WindowsService1\bin\Debug\WindowsService1.exe"
```

- Press Enter.
- It shows the following message:

The transacted install has completed.

Starting the windows service

- Go to “Start” – “Control Panel” – “System and Security” – “Administrative Tools” – “Services”.
- It shows “Automatic Emails Service”. Right click on the “Automatic Emails Service” windows service and click on “Start”. The windows service will be started within few seconds.
- To check whether it is working correctly or not:
 - Open gmail with following gmail id and password:

- Gmail id: harshadotnettraining2@gmail.com
 - Password: training123#
- Check the inbox. It gets a mail for every 30 seconds.

Stopping the windows service

- Go to “Start” – “Control Panel” – “System and Security” – “Administrative Tools” – “Services”.
- It shows “Automatic Emails Service”. Right click on the “Automatic Emails Service” windows service and click on “Stop”. The windows service will be stopped within few seconds.

Un-installing the windows service

- Go to “Start” – “All Programs” – “Visual Studio 2015” – Right click on “Developer Command Prompt for VS 2015” and click on “Run as administrator”. Click on “Yes”.
- Type the following command in the console window:

```
installutil/u "C:\CSharp\WindowsService1\WindowsService1\bin\Debug\WindowsService1.exe"
```

- Press Enter.
- It shows the following message:

The uninstall has completed.

C#.NET – WinForms – System.IO Namespace

The “System.IO.FileInfo” class - Example

Creating Project

- Create the following folders and files:
 - C:\CSharp
 - Sample.pdf
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FileInfoExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FileInfoExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace FileInfoExample
{
```

```
public partial class Form1 : Form
{
    Label label1, label2, label3, label4, label5, label6, label7, label8, label9,
    label10, label11;

    public Form1()
    {
        InitializeComponent();

        /* form properties */
        this.Font = new Font("Tahoma", 20);
        this.WindowState = FormWindowState.Maximized;

        /* label1 */
        label1 = new Label();
        label1.AutoSize = true;
        label1.Text = "label1";
        label1.Location = new Point(100, 50);
        this.Controls.Add(label1);

        /* label2 */
        label2 = new Label();
        label2.AutoSize = true;
        label2.Text = "label2";
        label2.Location = new Point(100, 100);
        this.Controls.Add(label2);

        /* label3 */
        label3 = new Label();
        label3.AutoSize = true;
        label3.Text = "label3";
        label3.Location = new Point(100, 150);
        this.Controls.Add(label3);

        /* label4 */
        label4 = new Label();
        label4.AutoSize = true;
```

```
label4.Text = "label4";
label4.Location = new Point(100, 200);
this.Controls.Add(label4);

/* label5 */
label5 = new Label();
label5.AutoSize = true;
label5.Text = "label5";
label5.Location = new Point(100, 250);
this.Controls.Add(label5);

/* label6 */
label6 = new Label();
label6.AutoSize = true;
label6.Text = "label6";
label6.Location = new Point(100, 300);
this.Controls.Add(label6);

/* label7 */
label7 = new Label();
label7.AutoSize = true;
label7.Text = "label7";
label7.Location = new Point(100, 350);
this.Controls.Add(label7);

/* label8 */
label8 = new Label();
label8.AutoSize = true;
label8.Text = "label8";
label8.Location = new Point(100, 400);
this.Controls.Add(label8);

/* label9 */
label9 = new Label();
label9.AutoSize = true;
label9.Text = "label9";
label9.Location = new Point(100, 450);
```

```
this.Controls.Add(label9);

/* label10 */
label10 = new Label();
label10.AutoSize = true;
label10.Text = "label10";
label10.Location = new Point(100, 500);
this.Controls.Add(label10);

/* label11 */
label11 = new Label();
label11.AutoSize = true;
label11.Text = "label11";
label11.Location = new Point(100, 550);
this.Controls.Add(label11);

/* FileInfo */
FileInfo finfo = new FileInfo(@"C:\CSharp\Sample.pdf");
string a = finfo.DirectoryName;
string b = finfo.Name;
string c = finfo.Extension;
string d = finfo.FullName;
bool e = finfo.Exists;
long f = finfo.Length;
DateTime g = finfo.CreationTime;
DateTime h = finfo.LastAccessTime;
DateTime i = finfo.LastWriteTime;
string j = finfo.Attributes.ToString();
bool k = finfo.IsReadOnly;

/* display output in labels */
label1.Text = "Directory name: " + a;
label2.Text = "Name: " + b;
label3.Text = "Extension: " + c;
label4.Text = "Full Name: " + d;
label5.Text = "Exists: " + Convert.ToString(e);
label6.Text = "Length: " + Convert.ToString(f) + " bytes";
```

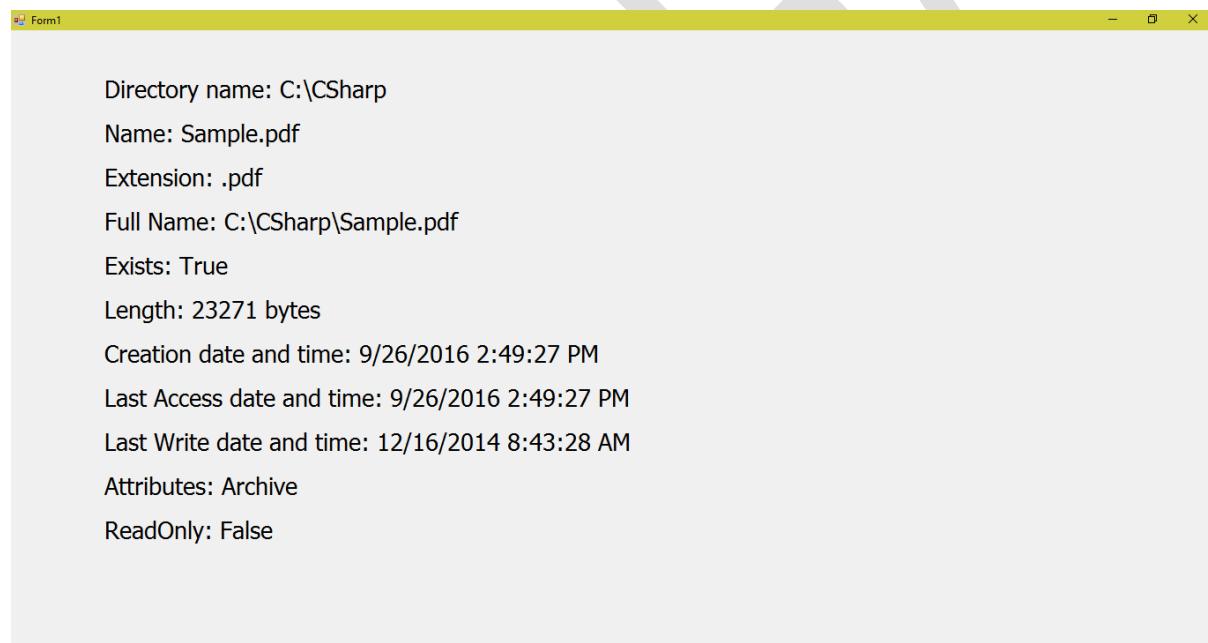
```

label7.Text = "Creation date and time: " + Convert.ToString(g);
label8.Text = "Last Access date and time: " + Convert.ToString(h);
label9.Text = "Last Write date and time: " + Convert.ToString(i);
label10.Text = "Attributes: " + j;
label11.Text = "ReadOnly: " + Convert.ToString(k);
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

It shows the file details as above.

The “System.IO.DirectoryInfo” class - Example**Creating Project**

- Create the following folders and files:

- C:\CSharp
 - sample
 - firstfolder (folder)
 - secondfolder (folder)
 - thirdfolder (folder)
 - New Text Document.txt
 - New Microsoft Word Document.docx

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DirectoryInfoExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DirectoryInfoExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace DirectoryInfoExample
{
    public partial class Form1 : Form
    {
```

```
Label label1, label2, label3, label4, label5, label6, label7, label8, label9,  
label10, label11;
```

```
public Form1()  
{  
    InitializeComponent();  
  
    /* form properties */  
    this.Font = new Font("Tahoma", 20);  
    this.WindowState = FormWindowState.Maximized;  
  
    /* label1 */  
    label1 = new Label();  
    label1.AutoSize = true;  
    label1.Text = "label1";  
    label1.Location = new Point(100, 50);  
    this.Controls.Add(label1);  
  
    /* label2 */  
    label2 = new Label();  
    label2.AutoSize = true;  
    label2.Text = "label2";  
    label2.Location = new Point(100, 100);  
    this.Controls.Add(label2);  
  
    /* label3 */  
    label3 = new Label();  
    label3.AutoSize = true;  
    label3.Text = "label3";  
    label3.Location = new Point(100, 150);  
    this.Controls.Add(label3);  
  
    /* label4 */  
    label4 = new Label();  
    label4.AutoSize = true;  
    label4.Text = "label4";  
    label4.Location = new Point(100, 200);
```

```
this.Controls.Add(label4);

/* label5 */
label5 = new Label();
label5.AutoSize = true;
label5.Text = "label5";
label5.Location = new Point(100, 250);
this.Controls.Add(label5);

/* label6 */
label6 = new Label();
label6.AutoSize = true;
label6.Text = "label6";
label6.Location = new Point(100, 300);
this.Controls.Add(label6);

/* label7 */
label7 = new Label();
label7.AutoSize = true;
label7.Text = "label7";
label7.Location = new Point(100, 350);
this.Controls.Add(label7);

/* label8 */
label8 = new Label();
label8.AutoSize = true;
label8.Text = "label8";
label8.Location = new Point(100, 400);
this.Controls.Add(label8);

/* label9 */
label9 = new Label();
label9.AutoSize = true;
label9.Text = "label9";
label9.Location = new Point(100, 450);
this.Controls.Add(label9);
```

```
/* label10 */
label10 = new Label();
label10.AutoSize = true;
label10.Text = "label10";
label10.Location = new Point(100, 500);
this.Controls.Add(label10);

/* label11 */
label11 = new Label();
label11.AutoSize = true;
label11.Text = "label11";
label11.Location = new Point(100, 550);
this.Controls.Add(label11);

/* DirectoryInfo */
DirectoryInfo dinfo = new DirectoryInfo(@"C:\CSharp\myfolder");
string a = dinfo.Parent.FullName;
string b = dinfo.Name;
string c = dinfo.FullName;
string d = dinfo.Root.FullName;
bool e = dinfo.Exists;
DateTime f = dinfo.CreationTime;
DateTime g = dinfo.LastAccessTime;
DateTime h = dinfo.LastWriteTime;
string i = dinfo.Attributes.ToString();
string temp1 = "", temp2 = "";
FileInfo[] files = dinfo.GetFiles();
DirectoryInfo[] directories = dinfo.GetDirectories();
foreach (FileInfo file in files)
{
    temp1 = temp1 + file.Name + ", ";
}
foreach (DirectoryInfo directory in directories)
{
    temp2 = temp2 + directory.Name + ", ";
}
```

```

/* display output in labels */
label1.Text = "Directory name: " + a;
label2.Text = "Name: " + b;
label3.Text = "Extension: " + c;
label4.Text = "Full Name: " + d;
label5.Text = "Exists: " + Convert.ToString(e);
label6.Text = "Creation date and time: " + Convert.ToString(f);
label7.Text = "Last Access date and time: " + Convert.ToString(g);
label8.Text = "Last Write date and time: " + Convert.ToString(h);
label9.Text = "Attributes: " + i;
label10.Text = "Files: " + temp1;
label11.Text = "Sub directories: " + temp2;
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



The screenshot shows a Windows application window titled "Form1". The window contains the following text output:

```

Directory name: C:\CSharp
Name: myfolder
Extension: C:\CSharp\myfolder
Full Name: C:\myfolder
Exists: True
Creation date and time: 9/26/2016 2:51:08 PM
Last Access date and time: 9/26/2016 2:51:08 PM
Last Write date and time: 9/26/2016 2:51:08 PM
Attributes: Directory
Files: file1.txt, file2.txt, file3.txt,
Sub directories: subfolder1, subfolder2, subfolder3,

```

It shows the folder details as above.

The “System.IO.Directory” class - Example

Creating Project

- Create the following folders:
 - C:\CSharp
 - Sample (folder)

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DirectoryExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DirectoryExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace DirectoryExample
{
    public partial class Form1 : Form
    {
        Button button1, button2, button3;
        string folderpath = @"C:\CSharp\Sample";
    }
}

```

```
public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(700, 400);

    /* button1 */
    button1 = new Button();
    button1.AutoSize = true;
    button1.Text = "Create";
    button1.Location = new Point(50, 150);
    button1.Click += Button1_Click;
    this.Controls.Add(button1);

    /* button2 */
    button2 = new Button();
    button2.AutoSize = true;
    button2.Text = "Delete";
    button2.Location = new Point(200, 150);
    button2.Click += Button2_Click;
    this.Controls.Add(button2);

    /* button3 */
    button3 = new Button();
    button3.AutoSize = true;
    button3.Text = "Move";
    button3.Location = new Point(350, 150);
    button3.Click += Button3_Click;
    this.Controls.Add(button3);
}

private void Button1_Click(object sender, EventArgs e)
{
    /* Create Folder */
}
```

```
if (Directory.Exists(folderpath) == true)
{
    MessageBox.Show("Folder already exists");
}
else
{
    Directory.CreateDirectory(folderpath);
    MessageBox.Show("Folder created");
}

private void Button2_Click(object sender, EventArgs e)
{
    /* Delete Folder */
    if (Directory.Exists(folderpath) == true)
    {
        Directory.Delete(folderpath, true);
        MessageBox.Show("Folder deleted");
    }
    else
    {
        MessageBox.Show("Folder not exists. Create the folder first.");
    }
}

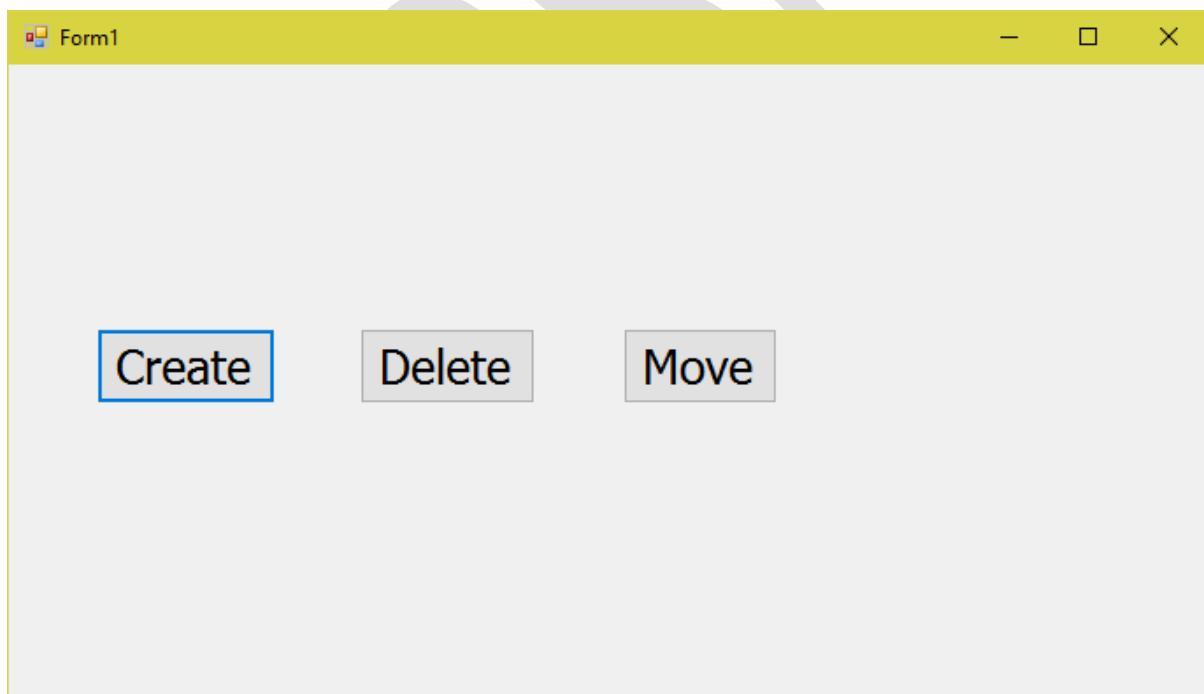
private void Button3_Click(object sender, EventArgs e)
{
    /* Move Folder */
    if (Directory.Exists(folderpath) == true)
    {
        if (Directory.Exists(@"C:\CSharp\folder2") == false)
        {
            Directory.CreateDirectory(@"C:\CSharp\folder2");
        }
        if (Directory.Exists(@"C:\CSharp\folder2\sample") == true)
        {
            Directory.Delete(@"C:\CSharp\folder2\sample");
        }
    }
}
```

```
        }
        Directory.Move(folderpath, @"C:\CSharp\folder2\sample");
        MessageBox.Show("Folder moved");
    }
    else
    {
        MessageBox.Show("Folder not exists. Create the Folder first.");
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Create”, “Delete”, “Move” buttons.

The “System.IO.File” class - Example

Creating Project

- Create the following folders and files:
 - C:\CSharp
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FileExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “FileExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace FileExample
{
    public partial class Form1 : Form
    {
        Button button1, button2, button3, button4;
        string filepath = @"C:\CSharp\Sample.txt";

        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(700, 400);
            /* button1 */
            button1 = new Button();
        }
    }
}
```

```
button1.AutoSize = true;
button1.Text = "Create";
button1.Location = new Point(50, 150);
button1.Click += Button1_Click;
this.Controls.Add(button1);
/* button2 */
button2 = new Button();
button2.AutoSize = true;
button2.Text = "Delete";
button2.Location = new Point(200, 150);
button2.Click += Button2_Click;
this.Controls.Add(button2);
/* button3 */
button3 = new Button();
button3.AutoSize = true;
button3.Text = "Move";
button3.Location = new Point(350, 150);
button3.Click += Button3_Click;
this.Controls.Add(button3);
/* button4 */
button4 = new Button();
button4.AutoSize = true;
button4.Text = "Copy";
button4.Location = new Point(500, 150);
button4.Click += Button4_Click;
this.Controls.Add(button4);
}
private void Button1_Click(object sender, EventArgs e)
{
/* Create File */
if (File.Exists(filepath) == true)
{
    MessageBox.Show("File already exists");
}
else
{
    FileStream fs = File.Create(filepath);
```

```
        fs.Close();
        MessageBox.Show("File created");
    }
}
private void Button2_Click(object sender, EventArgs e)
{
    /* Delete File */
    if (File.Exists(filepath) == true)
    {
        File.Delete(filepath);
        MessageBox.Show("File deleted");
    }
    else
    {
        MessageBox.Show("File not exists. Create the file first.");
    }
}
private void Button3_Click(object sender, EventArgs e)
{
    /* Move File */
    if (File.Exists(filepath) == true)
    {
        if (Directory.Exists(@"C:\CSharp\folder2") == false)
        {
            Directory.CreateDirectory(@"C:\CSharp\folder2");
        }
        if (File.Exists(@"C:\CSharp\folder2\sample.txt") == true)
        {
            File.Delete(@"C:\CSharp\folder2\sample.txt");
        }
        File.Move(filepath, @"C:\CSharp\folder2\sample.txt");
        MessageBox.Show("File moved");
    }
    else
    {
        MessageBox.Show("File not exists. Create the file first.");
    }
}
```

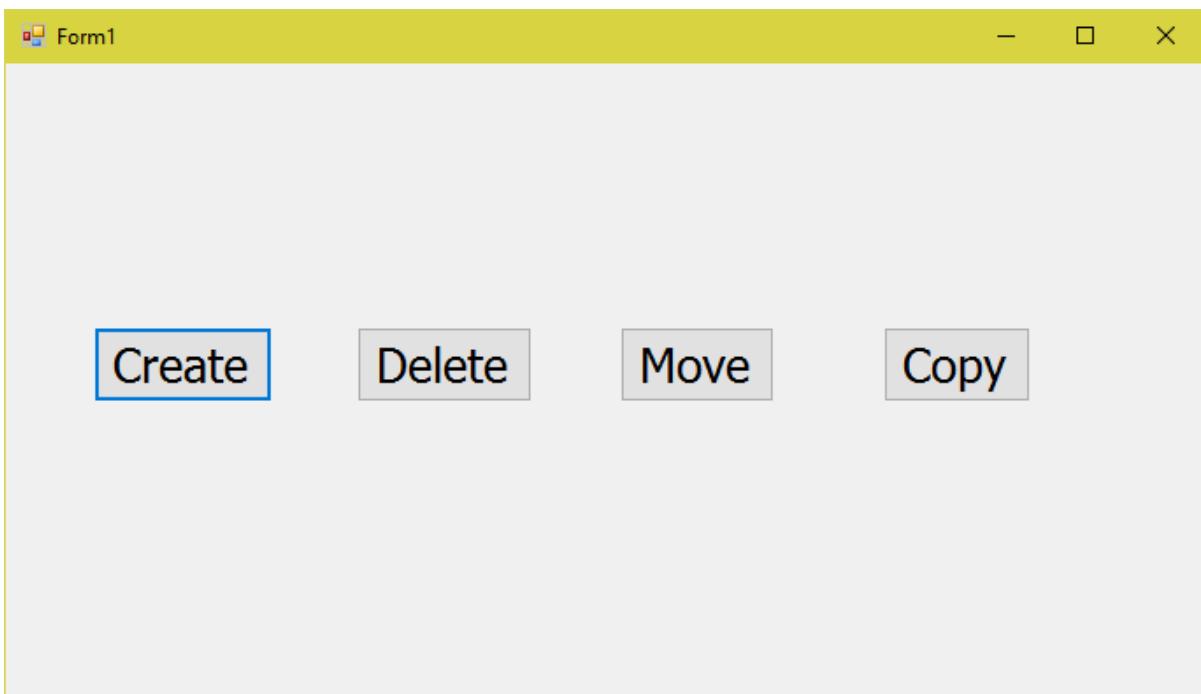
```
}

private void Button4_Click(object sender, EventArgs e)
{
    /* Copy File */
    if (File.Exists(filepath) == true)
    {
        if (Directory.Exists(@"C:\CSharp\folder2") == false)
        {
            Directory.CreateDirectory(@"C:\CSharp\folder2");
        }
        if (File.Exists(@"C:\CSharp\folder2\sample.txt") == true)
        {
            File.Delete(@"C:\CSharp\folder2\sample.txt");
        }
        File.Copy(filepath, @"C:\CSharp\folder2\sample.txt");
        MessageBox.Show("File copied");
    }
    else
    {
        MessageBox.Show("File not exists. Create the file first.");
    }
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Create”, “Delete”, “Move” and “Copy” buttons.

The “System.IO.StreamWriter” class - Example

Creating Project

- Create the following folder:
 - C:\CSharp
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “StreamWriterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StreamWriterExample”.

- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace StreamWriterExample
{
    public partial class Form1 : Form
    {
        Label label1;
        TextBox textbox1;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(900, 600);

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Enter your file content here:";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(800, 340);
            textbox1.Location = new Point(50, 100);
```

```
textbox1.Multiline = true;
this.Controls.Add(textbox1);

/* button1 */
button1 = new Button();
button1.Text = "Save";
button1.AutoSize = true;
button1.Location = new Point(50, 480);
button1.Click += Button1_Click;
this.Controls.Add(button1);

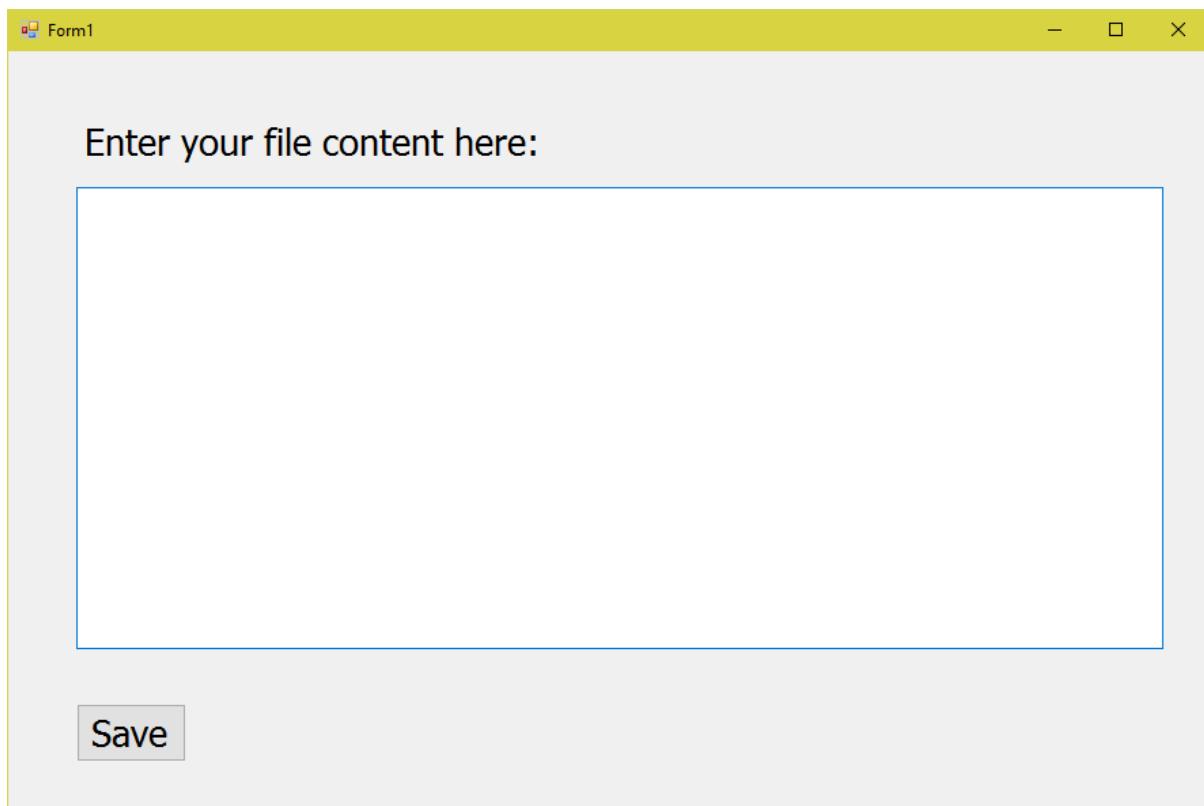
}

private void Button1_Click(object sender, EventArgs e)
{
    string filepath = @"C:\CSharp\sample.txt";
    FileInfo finfo = new FileInfo(filepath);
    if (finfo.Exists == true)
    {
        finfo.Delete();
    }
    FileStream fs = new FileStream(filepath, FileMode.Create,
FileAccess.Write);
    StreamWriter sw = new StreamWriter(fs);
    sw.WriteLine(textbox1.Text);
    sw.Close();
    MessageBox.Show("Saved");
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Type some content and click on “Save”.

The “System.IO.StreamReader” class - Example

Creating Project

- Create the following folder:
 - C:\CSharp
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “StreamReaderExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “StreamReaderExample”.

- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace StreamReaderExample
{
    public partial class Form1 : Form
    {
        Label label1;
        TextBox textbox1;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(900, 600);

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Enter your file content appears here:";
            label1.Location = new Point(50, 100);
            this.Controls.Add(label1);

            /* textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(800, 340);
            textbox1.Location = new Point(50, 150);
```

```
textbox1.Multiline = true;
this.Controls.Add(textbox1);

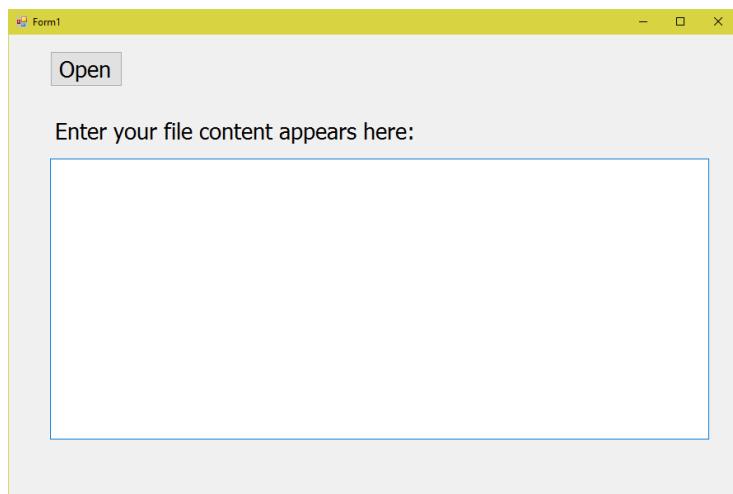
/* button1 */
button1 = new Button();
button1.Text = "Open";
button1.AutoSize = true;
button1.Location = new Point(50, 20);
button1.Click += Button1_Click;
this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
{
    string filepath = @"C:\CSharp\sample.txt";
    FileInfo finfo = new FileInfo(filepath);
    if (finfo.Exists == true)
    {
        FileStream fs = new FileStream(filepath, FileMode.Open,
FileAccess.Read);
        StreamReader sr = new StreamReader(fs);
        string s = sr.ReadToEnd();
        textbox1.Text = s;
        sr.Close();
        MessageBox.Show("Data loaded from the file successfully.");
    }
    else
    {
        MessageBox.Show("File not found");
    }
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Open”. It shows content from “C:\CSharp\sample.txt” file.

C#.NET – WinForms – ADO.NET

SqlConnection – Windows Authentication – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “WindowsAuthExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “WindowsAuthExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace WindowsAuthExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "SqlConnection - Windows Authentication";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Connect to SQL Server";
            button1.AutoSize = true;
            button1.Location = new Point(100, 100);
```

```
button1.Click += button1_Click;
this.Controls.Add(button1);
}

private void button1_Click(object sender, EventArgs e)
{
    //create reference variable
    SqlConnection cn;

    //create object
    cn = new SqlConnection();

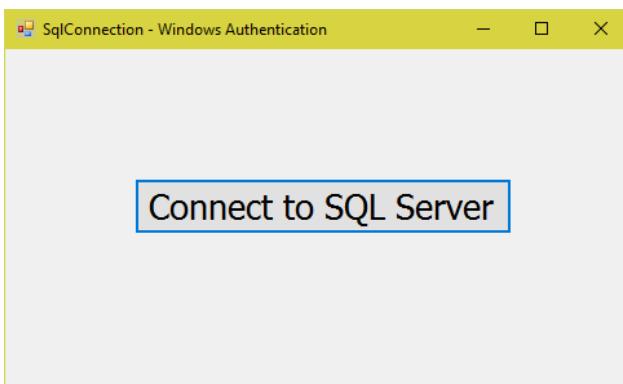
    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";

    //calling methods
    MessageBox.Show(cn.State.ToString()); //Output: Closed
    cn.Open();
    MessageBox.Show(cn.State.ToString()); //Output: Open
    cn.Close();
    MessageBox.Show(cn.State.ToString()); //Output: Closed
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Connect to SQL Server”.

Note: If any database connection problem, it shows exception (run time error).

SqlConnection – SQL Server Authentication – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

HARSHA

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SqlServerAuthExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerAuthExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```

using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace SqlServerAuthExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "SqlConnection - SqlServer Authentication";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Connect to SQL Server";
        }
    }
}

```

```
button1.AutoSize = true;
button1.Location = new Point(100, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);

}

private void button1_Click(object sender, EventArgs e)
{
    //create reference variable
    SqlConnection cn;

    //create object
    cn = new SqlConnection();

    //calling properties
    cn.ConnectionString = "data source=localhost; user id=sa;
password=123; initial catalog=company";

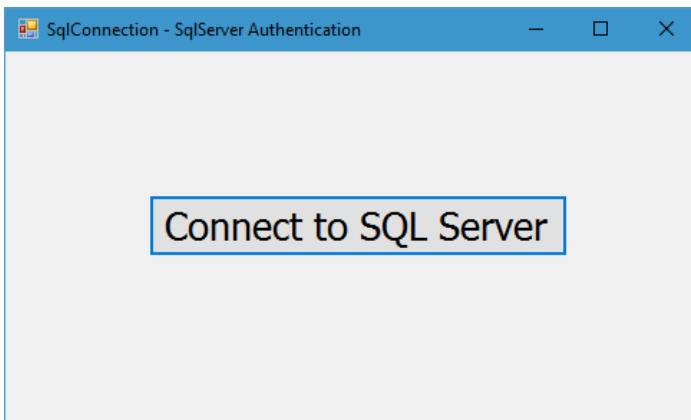
    //calling methods
    MessageBox.Show(cn.State.ToString()); //Output: Closed
    cn.Open();
    MessageBox.Show(cn.State.ToString()); //Output: Open
    cn.Close();
    MessageBox.Show(cn.State.ToString()); //Output: Closed
}

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Connect to SQL Server”.

Note: If any database connection problem, it shows exception (run time error).

SqlCommand – ExecuteScalar – Example

Creating Database

- **Note:** Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ExecuteScalarExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExecuteScalarExample”. Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace ExecuteScalarExample
{
    public partial class Form1 : Form
    {
        Button button1;
        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
```

```

this.Size = new Size(500, 300);
this.Text = "Execute Scalar";
this.StartPosition = FormStartPosition.CenterScreen;
/* button1 */
button1 = new Button();
button1.Text = "Execute Scalar";
button1.AutoSize = true;
button1.Location = new Point(130, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);
}

private void button1_Click(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select EmpName from Employees where
EmpID=1";
    cmd.Connection = cn;

    /* call methods */
    cn.Open();
    object obj = cmd.ExecuteScalar();
    cn.Close();
    string n = Convert.ToString(obj);
    string msg = "Emp Name: " + n;
    MessageBox.Show(msg);
}

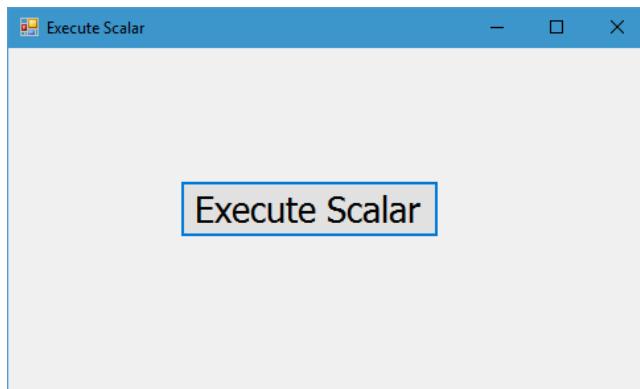
```

```
}
```

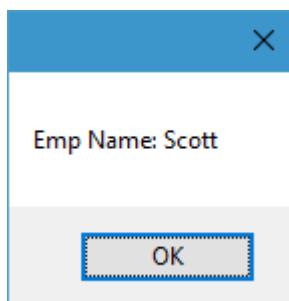
```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Click on “Execute Scalar”.



Note: If any database connection problem, it shows exception (run time error).

SqlCommand – ExecuteScalar – Example 2**Creating Database**

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ExecuteScalarExample2”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExecuteScalarExample2”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;  
using System.Windows.Forms;
```

```
using System.Drawing;
using System.Data.SqlClient;

namespace ExecuteScalarExample2
{
    public partial class Form1 : Form
    {
        Button button1;
        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "Execute Scalar";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Execute Scalar";
            button1.AutoSize = true;
            button1.Location = new Point(130, 100);
            button1.Click += button1_Click;
            this.Controls.Add(button1);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            /* create pointers */
            SqlConnection cn;
            SqlCommand cmd;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();

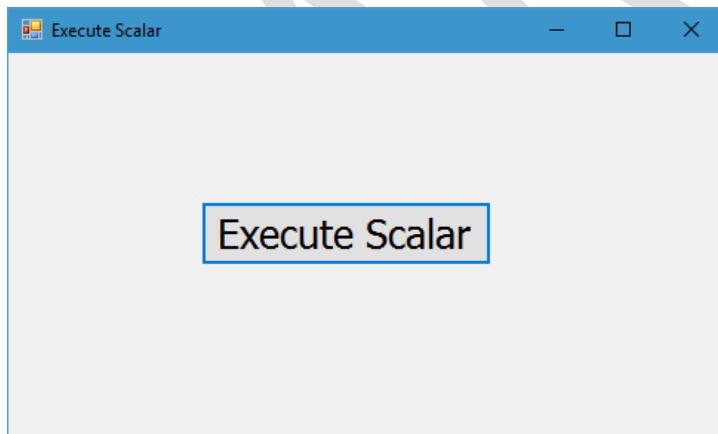
            /* call properties */
        }
    }
}
```

```
cn.ConnectionString = "data source=localhost; integrated  
security=yes; initial catalog=company";  
cmd.CommandText = "select count(*) from Employees";  
cmd.Connection = cn;  
  
/* call methods */  
cn.Open();  
object obj = cmd.ExecuteScalar();  
cn.Close();  
string n = Convert.ToString(obj);  
string msg = "Emps Count: " + n;  
MessageBox.Show(msg);  
}  
}  
}
```

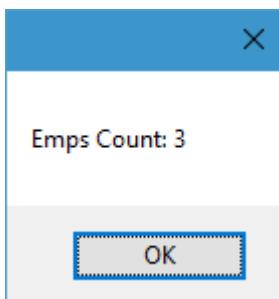
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Execute Scalar”.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Connection Oriented Model – Single Record – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
    EmpID int primary key,
    EmpName nvarchar(max),
    Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “COMSingleRecordExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMSingleRecordExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace COMSingleRecordExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
```

```
this.Text = "Connection Oriented Model - Single Record";
this.StartPosition = FormStartPosition.CenterScreen;
this.Load += Form1_Load;
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees where empid=1";
    cmd.Connection = cn;

    /* call methods */
    cn.Open();
    dr = cmd.ExecuteReader();
    if (dr.Read())
    {
        object obj1, obj2, obj3;

        obj1 = dr["EmpID"];
        obj2 = dr["EmpName"];
        obj3 = dr["Salary"];

        int eid;
        string ename;
        decimal sal;
```

```
eid = Convert.ToInt32(obj1);
ename = Convert.ToString(obj2);
sal = Convert.ToDecimal(obj3);

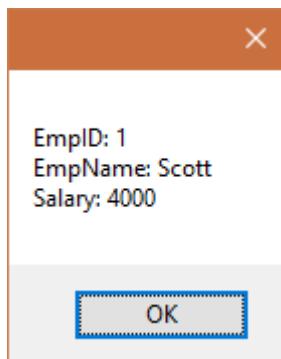
MessageBox.Show("EmpID: " + eid + "\nEmpName: " + ename +
"\nSalary: " + sal);
}

cn.Close();
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on OK.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Connection Oriented Model – Multiple Records – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

go

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

HARSHA

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “COMMmultipleRecordsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMMmultipleRecordsExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
namespace COMMmultipleRecordsExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "Connection Oriented Model - Multiple Records";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
        }
}
```

```

button1 = new Button();
button1.Text = "Connection Oriented Model";
button1.AutoSize = true;
button1.Location = new Point(70, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);
}
private void button1_Click(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;
    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees";
    cmd.Connection = cn;
    /* call methods */
    cn.Open();
    dr = cmd.ExecuteReader();
    while (dr.Read())
    {
        object obj1, obj2, obj3;
        obj1 = dr["EmplID"];
        obj2 = dr["EmpName"];
        obj3 = dr["Salary"];
        int eid;
        string ename;
        decimal sal;
        eid = Convert.ToInt32(obj1);
        ename = Convert.ToString(obj2);
        sal = Convert.ToDecimal(obj3);
        string msg = "EmplID: " + eid + "\nEmpName: " + ename + "\nSalary: " + sal;
    }
}

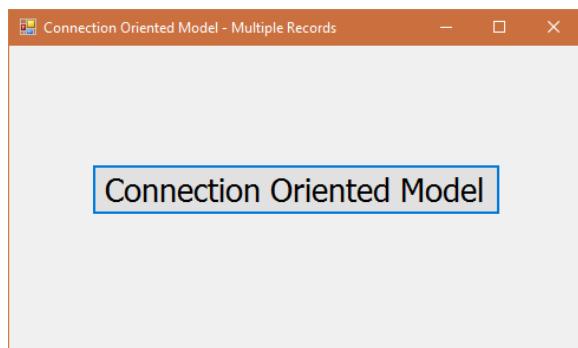
```

```
    MessageBox.Show(msg);
}
cn.Close();
}
}
}
```

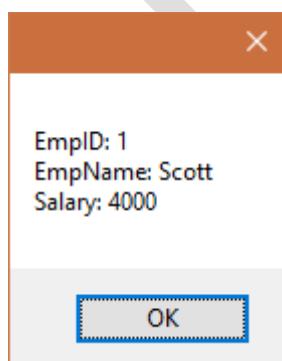
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

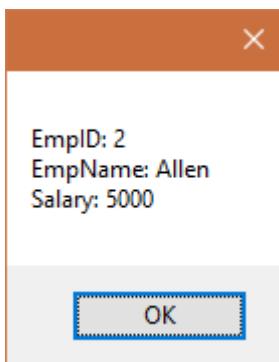
Output



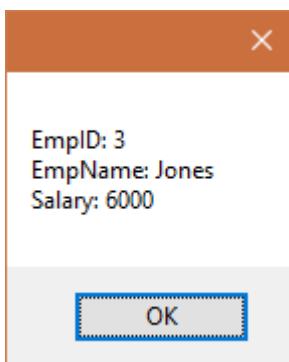
Click on “Connection Oriented Model”.



Click on OK.



Click on OK.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Connection Oriented Model – Multiple Records - Label – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database company

go

```
use company
```

```
go
```

```
create table Employees(  

EmpID int primary key,  

EmpName nvarchar(max),  

Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  

insert into Employees values(2, 'Allen', 5000)  

insert into Employees values(3, 'Jones', 6000)  

go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “COMMmultipleRecordsLabelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMMmultipleRecordsLabelExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;  

using System.Windows.Forms;  

using System.Drawing;  

using System.Data.SqlClient;
```

```
namespace COMMultipleRecordsLabelExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "Connection Oriented Model";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.AutoScroll = true;
            this.Load += Form1_Load;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataReader dr;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();

            /* call properties */
            cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
            cmd.CommandText = "select * from Employees";
            cmd.Connection = cn;

            /* call methods */
            cn.Open();
        }
    }
}
```

```
dr = cmd.ExecuteReader();
int n = 50;
while (dr.Read())
{
    object obj1, obj2, obj3;

    obj1 = dr["EmpID"];
    obj2 = dr["EmpName"];
    obj3 = dr["Salary"];

    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    Label label1, label2, label3;
    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = Convert.ToString(eid);
    label1.Location = new Point(50, n);
    this.Controls.Add(label1);
    /* label2 */
    label2 = new Label();
    label2.AutoSize = true;
    label2.Text = ename;
    label2.Location = new Point(150, n);
    this.Controls.Add(label2);
    /* label3 */
    label3 = new Label();
    label3.AutoSize = true;
    label3.Text = Convert.ToString(sal);
    label3.Location = new Point(350, n);
    this.Controls.Add(label3);
    /* go to next row */
    n += 50;
```

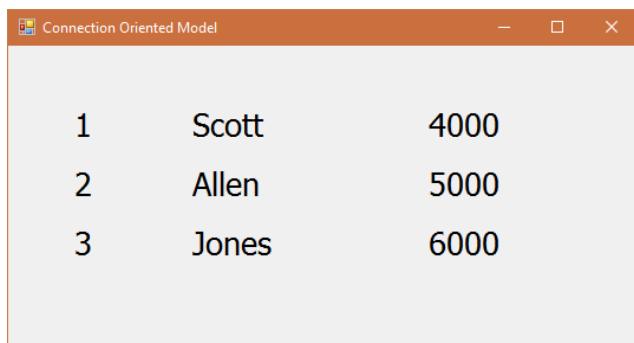
```

        }
        cn.Close();
    }
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output


The screenshot shows a Windows application window titled "Connection Oriented Model". Inside the window, there is a table with three columns and three rows of data:

1	Scott	4000
2	Allen	5000
3	Jones	6000

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Connection Oriented Model – SqlParameter – Example**Creating Database**

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database company

go

use company

go

```
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “COMSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMSqlParameterExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
```

```
namespace COMSqlParameterExample
```

```
{  
    public partial class Form1 : Form  
    {  
        Label label1, label2, label3;  
        TextBox textbox1, textbox2, textbox3;  
        Button button1;  
  
        public Form1()  
        {  
            InitializeComponent();  
  
            /* form properties */  
            this.Font = new Font("Tahoma", 20);  
            this.Size = new Size(600, 470);  
            this.Text = "Connection Oriented Model - with SqlParameter";  
            this.StartPosition = FormStartPosition.CenterScreen;  
  
            /* label1 */  
            label1 = new Label();  
            label1.AutoSize = true;  
            label1.Text = "Emp ID: ";  
            label1.Location = new Point(50, 50);  
            this.Controls.Add(label1);  
  
            /* label2 */  
            label2 = new Label();  
            label2.AutoSize = true;  
            label2.Text = "Emp Name: ";  
            label2.Location = new Point(50, 200);  
            this.Controls.Add(label2);  
  
            /* label3 */  
            label3 = new Label();  
            label3.AutoSize = true;  
            label3.Text = "Salary: ";  
            label3.Location = new Point(50, 300);  
            this.Controls.Add(label3);  
    }  
}
```

```
/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(200, 50);
textbox1.Location = new Point(250, 50);
this.Controls.Add(textbox1);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "OK";
button1.Location = new Point(250, 100);
button1.Click += Button1_Click;
this.AcceptButton = button1;
this.Controls.Add(button1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(200, 50);
textbox2.Location = new Point(250, 200);
textbox2.ReadOnly = true;
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(200, 50);
textbox3.Location = new Point(250, 300);
textbox3.ReadOnly = true;
this.Controls.Add(textbox3);
}

private void Button1_Click(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1;
```

```

SqlDataReader dr;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
p1 = new SqlParameter();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees where
empid=@EmpID";
cmd.Connection = cn;
p1.ParameterName = "@EmpID";
p1.Value = textBox1.Text;
cmd.Parameters.Add(p1);

/* call methods */
cn.Open();
dr = cmd.ExecuteReader();
if (dr.Read())
{
    object obj1, obj2;
    obj1 = dr["EmpName"];
    obj2 = dr["Salary"];
    string ename;
    decimal sal;
    ename = Convert.ToString(obj1);
    sal = Convert.ToDecimal(obj2);
    textBox2.Text = ename;
    textBox3.Text = Convert.ToString(sal);
}
else
{
    textBox2.Text = "";
    textBox3.Text = "";
    MessageBox.Show("No data found");
}

```

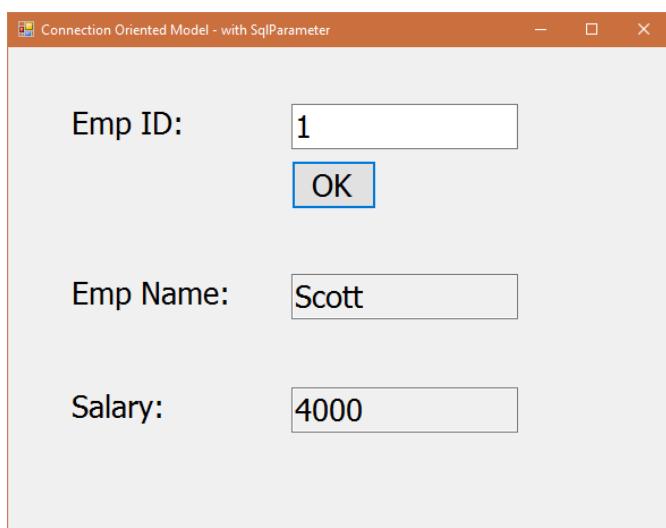
```

        }
        cn.Close();
    }
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Enter EmpID as “1” and Click on OK.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Connection Oriented Model – SqlParameter – ComboBox – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.

- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “COMWithComboBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “COMWithComboBoxExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace COMWithComboBoxExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        ComboBox combobox1;
        TextBox textbox1, textbox2;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(600, 470);
            this.Text = "Connection Oriented Model - with SqlParameter";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Emp ID: ";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* label2 */
            label2 = new Label();
            label2.AutoSize = true;
            label2.Text = "Emp Name: ";
            label2.Location = new Point(50, 150);
            this.Controls.Add(label2);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Load data from database
        }
    }
}
```

```
/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary: ";
label3.Location = new Point(50, 250);
this.Controls.Add(label3);

/* combobox1 */
comboBox1 = new ComboBox();
comboBox1.Size = new Size(200, 50);
comboBox1.Location = new Point(250, 50);
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox1.SelectedIndexChanged +=
    Combobox1_SelectedIndexChanged;
this.Controls.Add(comboBox1);

/* textbox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(200, 50);
textBox1.ReadOnly = true;
textBox1.Location = new Point(250, 150);
this.Controls.Add(textBox1);

/* textbox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(200, 50);
textBox2.Location = new Point(250, 250);
textBox2.ReadOnly = true;
this.Controls.Add(textBox2);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
```

```

SqlDataReader dr;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;

/* call methods */
cn.Open();
dr = cmd.ExecuteReader();
while (dr.Read())
{
    object obj1;
    obj1 = dr["EmpID"];
    int eid;
    eid = Convert.ToInt32(obj1);
    combobox1.Items.Add(eid);
}
cn.Close();
}

private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1;
    SqlDataReader dr;

    /* create objects */
    cn = new SqlConnection();

```

```
cmd = new SqlCommand();
p1 = new SqlParameter();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees where
empid=@EmpID";
cmd.Connection = cn;
p1.ParameterName = "@EmpID";
p1.Value = combobox1.SelectedItem;
cmd.Parameters.Add(p1);

/* call methods */
cn.Open();
dr = cmd.ExecuteReader();
if (dr.Read())
{
    object obj1, obj2;

    obj1 = dr["EmpName"];
    obj2 = dr["Salary"];

    string ename;
    decimal sal;

    ename = Convert.ToString(obj1);
    sal = Convert.ToDecimal(obj2);

    textbox1.Text = ename;
    textbox2.Text = Convert.ToString(sal);
}
else
{
    textbox1.Text = "";
    textbox2.Text = "";
    MessageBox.Show("No data found");
}
```

```
        }
```



```
    cn.Close();
```

```
    }
```

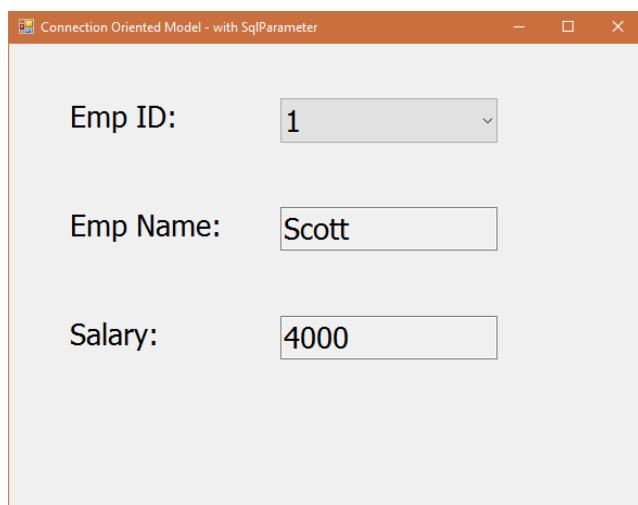
```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



All the EmpID's appear in the ComboBox.

Select EmpID and it shows EmpName and Salary.

Note: If any database connection problem, it shows exception (run time error).

DataSet - Example

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
 - Select “.NET Framework 4.8”. Select “Visual C#”.
 - Select “Windows Forms Application”.

- Type the project name as “DisconnectedModelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DisconnectedModelExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
/* Expected data:
```

Categories:

CategoryID	CategoryName
1	ab
2	cd

Products:

ProductID	ProductName	Cost
101	prod1	400
102	prod2	500
103	prd3	600

```
*/
```

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data;
```

```
namespace DataSetExample
```

```
{
    public partial class Form1 : Form
    {
        Button button1;
        public Form1()
        {
            InitializeComponent();
            /* form properties */
        }
    }
}
```

```

this.Font = new Font("Tahoma", 20);
this.Size = new Size(500, 300);
this.Text = "DataSet";
this.StartPosition = FormStartPosition.CenterScreen;
/* button1 */
button1 = new Button();
button1.Text = "Create DataSet";
button1.AutoSize = true;
button1.Location = new Point(140, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);
}
private void button1_Click(object sender, EventArgs e)
{
    //creating dataset
    DataSet ds = new DataSet();

    //creating 2 tables
    DataTable dt1 = new DataTable() { TableName = "Categories" };
    DataTable dt2 = new DataTable() { TableName = "Products" };

    //creating 2 columns for table1
    DataColumn col1 = new DataColumn() { ColumnName = "CategoryID",
    DataType = typeof(int) };
    DataColumn col2 = new DataColumn() { ColumnName =
    "CategoryName", DataType = typeof(string) };

    //creating 3 columns for table2
    DataColumn col3 = new DataColumn() { ColumnName = "ProductID",
    DataType = typeof(int) };
    DataColumn col4 = new DataColumn() { ColumnName =
    "ProductName", DataType = typeof(string) };
    DataColumn col5 = new DataColumn() { ColumnName = "Cost",
    DataType = typeof(decimal) };

    //adding columns to table1
    dt1.Columns.Add(col1);
}

```

```
dt1.Columns.Add(col2);

//adding columns to table2
dt2.Columns.Add(col3);
dt2.Columns.Add(col4);
dt2.Columns.Add(col5);

//creating 2 rows for table1
DataRow drow1 = dt1.NewRow();
DataRow drow2 = dt1.NewRow();

//creating 3 rows for table2
DataRow drow3 = dt2.NewRow();
DataRow drow4 = dt2.NewRow();
DataRow drow5 = dt2.NewRow();

//adding rows to table1
dt1.Rows.Add(drow1);
dt1.Rows.Add(drow2);

//adding rows to table2
dt2.Rows.Add(drow3);
dt2.Rows.Add(drow4);
dt2.Rows.Add(drow5);

//adding tables to dataset
ds.Tables.Add(dt1);
ds.Tables.Add(dt2);

***** setting data *****

//setting data in table1
dt1.Rows[0]["CategoryID"] = 1;
dt1.Rows[0]["CategoryName"] = "ab";
dt1.Rows[1]["CategoryID"] = 2;
dt1.Rows[1]["CategoryName"] = "cd";
```

```
//setting data in table2
dt2.Rows[0]["ProductID"] = 101;
dt2.Rows[0]["ProductName"] = "prod1";
dt2.Rows[0]["Cost"] = 400;
dt2.Rows[1]["ProductID"] = 102;
dt2.Rows[1]["ProductName"] = "prod2";
dt2.Rows[1]["Cost"] = 500;
dt2.Rows[2]["ProductID"] = 103;
dt2.Rows[2]["ProductName"] = "prod3";
dt2.Rows[2]["Cost"] = 600;

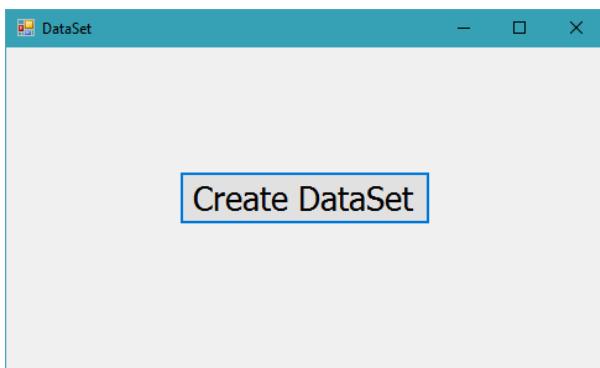
//getting data from table1
MessageBox.Show(dt1.TableName + ": ");
for (int i = 0; i < dt1.Rows.Count; i++)
{
    MessageBox.Show(dt1.Rows[i]["CategoryID"] + ", " +
dt1.Rows[i]["CategoryName"]);
}

//getting data from table2
MessageBox.Show(dt2.TableName + ": ");
for (int i = 0; i < dt2.Rows.Count; i++)
{
    MessageBox.Show(dt2.Rows[i]["ProductID"] + ", " +
dt2.Rows[i]["ProductName"] + ", " + dt2.Rows[i]["Cost"]);
}
```

Running the Project

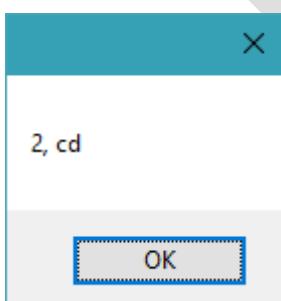
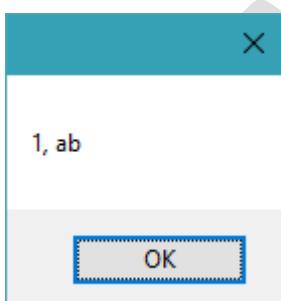
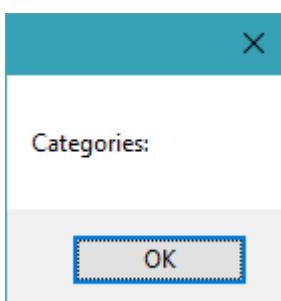
- Go to “Debug” menu and click on “Start Debugging”.

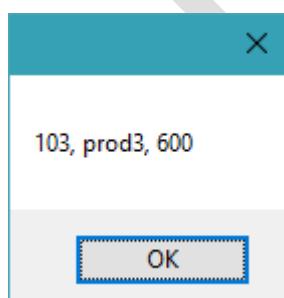
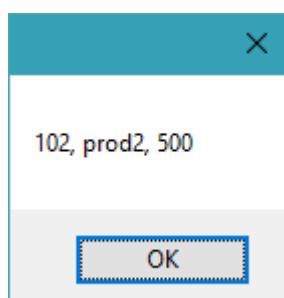
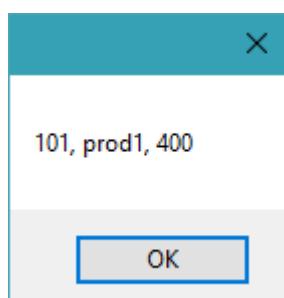
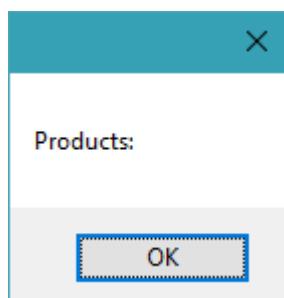
Output



Click on "Create DataSet".

It shows data:





ADO.NET Disconnected Model – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.

- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DisconnectedModelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DisconnectedModelExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;  
using System.Windows.Forms;
```

```
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DisconnectedModelExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "Disconnected Model";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.AutoScroll = true;
            this.Load += Form1_Load;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            /* create reference variables */
            SqlConnection cn;
            SqlCommand cmd;
            SqlDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            /* create objects */
            cn = new SqlConnection();
            cmd = new SqlCommand();
            adp = new SqlDataAdapter();
            ds = new DataSet();
        }
    }
}
```

```
/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
int n = 50;
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);

    Label label1, label2, label3;

    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = Convert.ToString(eid);
    label1.Location = new Point(50, n);
    this.Controls.Add(label1);
```

```

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = ename;
label2.Location = new Point(150, n);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = Convert.ToString(sal);
label3.Location = new Point(350, n);
this.Controls.Add(label3);

/* go to next row */
n += 50;
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Disconnected Model		
1	Scott	4000
2	Allen	5000
3	Jones	6000

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Multiple Tables - Example

Creating Database

- Note: Ignore this step, if you have created “departmentsandemployeesdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database departmentsandemployeesdatabase
go
```

```
use departmentsandemployeesdatabase
go
```

```
create table Departments(
DeptNo int primary key,
DeptName nvarchar(max),
Loc nvarchar(max))
go
```

```
create table Employees(
EmpID int primary key,
EmpName varchar(max),
Salary decimal,
DeptNo int references Departments(DeptNo))
go
```

```
insert into Departments values(10, 'Acounting', 'New York')
insert into Departments values(20, 'Operations', 'New Delhi')
insert into Departments values(30, 'Sales', 'New Jersy')
```

```
insert into Employees values(1, 'Scott', 3000, 10)
insert into Employees values(2, 'Allen', 6500, 10)
insert into Employees values(3, 'Jones', 4577, 20)
```

```
insert into Employees values(4, 'James', 9500, 20)
insert into Employees values(5, 'Smith', 3345, 30)
insert into Employees values(6, 'Harry', 2500, 30)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MultipleTablesExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MultipleTablesExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace MultipleTablesExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
```

```
InitializeComponent();
```

```
/* form properties */  
this.Font = new Font("Tahoma", 20);  
this.Size = new Size(700, 500);  
this.Text = "Disconnected Model";  
this.StartPosition = FormStartPosition.CenterScreen;  
this.AutoScroll = true;  
this.Load += Form1_Load;  
}
```

```
private void Form1_Load(object sender, EventArgs e)  
{  
    /* create reference variables */  
    SqlConnection cn;  
    SqlCommand cmd;  
    SqlDataAdapter adp;  
    DataSet ds;  
    DataTable dt1, dt2;  
    DataRow drow;  
    /* create objects */  
    cn = new SqlConnection();  
    cmd = new SqlCommand();  
    adp = new SqlDataAdapter();  
    ds = new DataSet();  
    /* call properties */  
    cn.ConnectionString = "data source=localhost; integrated  
security=yes; initial catalog=departmentsandemployeesdatabase";  
    cmd.CommandText = "select * from Departments select * from  
Employees";  
    cmd.Connection = cn;  
    adp.SelectCommand = cmd;  
    /* call methods */  
    adp.Fill(ds);  
    /* departments */  
    dt1 = ds.Tables[0];  
    int n = 50;
```

```
for (int i = 0; i < dt1.Rows.Count; i++)
{
    drow = dt1.Rows[i];
    object obj1, obj2, obj3;
    obj1 = drow["DeptNo"];
    obj2 = drow["DeptName"];
    obj3 = drow["Loc"];
    int dno;
    string dname;
    string loc;
    dno = Convert.ToInt32(obj1);
    dname = Convert.ToString(obj2);
    loc = Convert.ToString(obj3);
    Label label1, label2, label3;
    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = Convert.ToString(dno);
    label1.Location = new Point(50, n);
    this.Controls.Add(label1);
    /* label2 */
    label2 = new Label();
    label2.AutoSize = true;
    label2.Text = dname;
    label2.Location = new Point(150, n);
    this.Controls.Add(label2);
    /* label3 */
    label3 = new Label();
    label3.AutoSize = true;
    label3.Text = Convert.ToString(loc);
    label3.Location = new Point(350, n);
    this.Controls.Add(label3);
    /* go to next row */
    n += 50;
}

/* employees */
```

```
dt2 = ds.Tables[1];
n += 100;
for (int i = 0; i < dt2.Rows.Count; i++)
{
    drow = dt2.Rows[i];
    object obj1, obj2, obj3, obj4;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    obj4 = drow["DeptNo"];
    int eid;
    string ename;
    decimal sal;
    int dno;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    dno = Convert.ToInt32(obj4);

    Label label1, label2, label3, label4;
    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = Convert.ToString(eid);
    label1.Location = new Point(50, n);
    this.Controls.Add(label1);
    /* label2 */
    label2 = new Label();
    label2.AutoSize = true;
    label2.Text = ename;
    label2.Location = new Point(150, n);
    this.Controls.Add(label2);
    /* label3 */
    label3 = new Label();
    label3.AutoSize = true;
    label3.Text = Convert.ToString(sal);
    label3.Location = new Point(350, n);
```

```

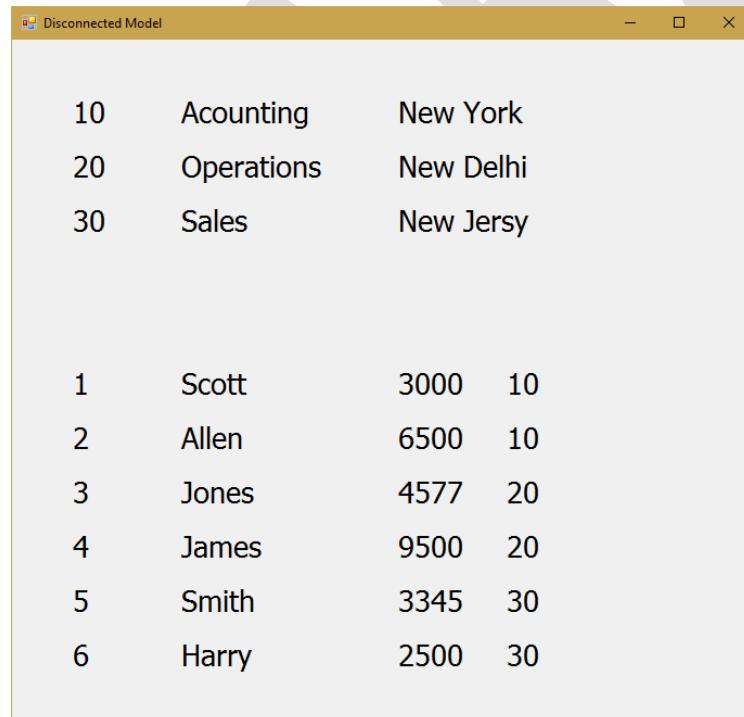
this.Controls.Add(label3);
/* label4 */
label4 = new Label();
label4.AutoSize = true;
label4.Text = Convert.ToString(dno);
label4.Location = new Point(450, n);
this.Controls.Add(label4);

/* go to next row */
n += 50;
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output


The screenshot shows a Windows application window titled "Disconnected Model". It contains two tables of data:

10	Accounting	New York		
20	Operations	New Delhi		
30	Sales	New Jersy		

1	Scott	3000	10	
2	Allen	6500	10	
3	Jones	4577	20	
4	James	9500	20	
5	Smith	3345	30	
6	Harry	2500	30	

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Joins - Example

Creating Database

- Note: Ignore this step, if you have created “departmentsandemployeesdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database departmentsandemployeesdatabase
```

```
go
```

```
use departmentsandemployeesdatabase
```

```
go
```

```
create table Departments(  
DeptNo int primary key,  
DeptName nvarchar(max),  
Loc nvarchar(max))
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName varchar(max),  
Salary decimal,  
DeptNo int references Departments(DeptNo))
```

```
go
```

```
insert into Departments values(10, 'Accounting', 'New York')  
insert into Departments values(20, 'Operations', 'New Delhi')  
insert into Departments values(30, 'Sales', 'New Jersey')
```

```
insert into Employees values(1, 'Scott', 3000, 10)
insert into Employees values(2, 'Allen', 6500, 10)
insert into Employees values(3, 'Jones', 4577, 20)
insert into Employees values(4, 'James', 9500, 20)
insert into Employees values(5, 'Smith', 3345, 30)
insert into Employees values(6, 'Harry', 2500, 30)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “JoinsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “JoinsExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace JoinsExample
{
    public partial class Form1 : Form
    {
```

```
public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(950, 500);
    this.Text = "Disconnected Model";
    this.StartPosition = FormStartPosition.CenterScreen;
    this.AutoScroll = true;
    this.Load += Form1_Load;
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;
    DataTable dt;
    DataRow drow;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=departmentsandemployeesdatabase";
    cmd.CommandText = "select * from employees inner join departments on employees.deptno=departments.deptno";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;
```

```

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
int n = 50;
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3, obj4, obj5, obj6;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
    obj4 = drow["DeptNo"];
    obj5 = drow["DeptName"];
    obj6 = drow["Loc"];

    int eid;
    string ename;
    decimal sal;
    int dno;
    string dname;
    string loc;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    dno = Convert.ToInt32(obj4);
    dname = Convert.ToString(obj5);
    loc = Convert.ToString(obj6);

    Label label1, label2, label3, label4, label5, label6;

    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = Convert.ToString(eid);
    label1.Location = new Point(50, n);
}

```

```
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = ename;
label2.Location = new Point(150, n);
this.Controls.Add(label2);
/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = Convert.ToString(sal);
label3.Location = new Point(350, n);
this.Controls.Add(label3);
/* label4 */
label4 = new Label();
label4.AutoSize = true;
label4.Text = Convert.ToString(dno);
label4.Location = new Point(450, n);
this.Controls.Add(label4);
/* label5 */
label5 = new Label();
label5.AutoSize = true;
label5.Text = Convert.ToString(dname);
label5.Location = new Point(550, n);
this.Controls.Add(label5);

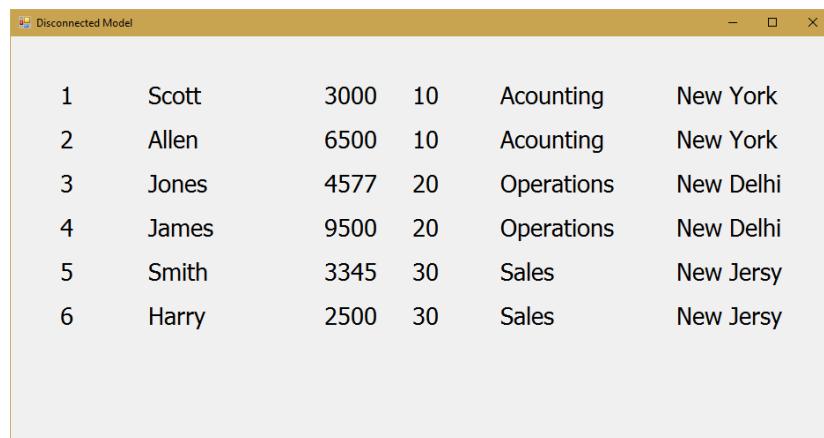
/* label6 */
label6 = new Label();
label6.AutoSize = true;
label6.Text = Convert.ToString(loc);
label6.Location = new Point(750, n);
this.Controls.Add(label6);
/* go to next row */
n += 50;
}
}
```

```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output


The screenshot shows a Windows application window titled "Disconnected Model". Inside the window, there is a grid table with the following data:

1	Scott	3000	10	Acouting	New York
2	Allen	6500	10	Acouting	New York
3	Jones	4577	20	Operations	New Delhi
4	James	9500	20	Operations	New Delhi
5	Smith	3345	30	Sales	New Jersy
6	Harry	2500	30	Sales	New Jersy

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Record Navigations - Example**Creating Database**

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database company

go

use company

go

```
create table Employees(
    EmpID int primary key,
    EmpName nvarchar(max),
    Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “RecordNavigationsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RecordNavigationsExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;
```

namespace RecordNavigationsExample

```
{  
    public partial class Form1 : Form  
    {  
        DataTable dt;  
        DataRow drow;  
        Label label1, label2, label3, label4;  
        TextBox textbox1, textbox2, textbox3, textbox4;  
        Button button1, button2, button3, button4;  
  
        public Form1()  
        {  
            InitializeComponent();  
  
            /* form properties */  
            this.Font = new Font("Tahoma", 20);  
            this.Size = new Size(650, 400);  
            this.Text = "Disconnected Model";  
            this.StartPosition = FormStartPosition.CenterScreen;  
            this.Load += Form1_Load;  
  
            /* label1 */  
            label1 = new Label();  
            label1.AutoSize = true;  
            label1.Text = "Record No: ";  
            label1.Location = new Point(50, 50);  
            this.Controls.Add(label1);  
  
            /* label2 */  
            label2 = new Label();  
            label2.AutoSize = true;  
            label2.Text = "Emp ID: ";  
            label2.Location = new Point(50, 100);  
            this.Controls.Add(label2);  
  
            /* label3 */  
            label3 = new Label();  
            label3.AutoSize = true;
```

```
label3.Text = "Emp Name:";  
label3.Location = new Point(50, 150);  
this.Controls.Add(label3);  
  
/* label4 */  
label4 = new Label();  
label4.AutoSize = true;  
label4.Text = "Salary:";  
label4.Location = new Point(50, 200);  
this.Controls.Add(label4);  
  
/* textbox1 */  
textbox1 = new TextBox();  
textbox1.Size = new Size(300, 50);  
textbox1.ReadOnly = true;  
textbox1.Location = new Point(250, 50);  
this.Controls.Add(textbox1);  
  
/* textbox2 */  
textbox2 = new TextBox();  
textbox2.Size = new Size(300, 50);  
textbox2.ReadOnly = true;  
textbox2.Location = new Point(250, 100);  
this.Controls.Add(textbox2);  
  
/* textbox3 */  
textbox3 = new TextBox();  
textbox3.Size = new Size(300, 50);  
textbox3.ReadOnly = true;  
textbox3.Location = new Point(250, 150);  
this.Controls.Add(textbox3);  
  
/* textbox4 */  
textbox4 = new TextBox();  
textbox4.Size = new Size(300, 50);  
textbox4.ReadOnly = true;  
textbox4.Location = new Point(250, 200);
```

```
this.Controls.Add(textBox4);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "First";
button1.Location = new Point(50, 280);
button1.Click += Button1_Click;
this.Controls.Add(button1);

/* button2 */
button2 = new Button();
button2.AutoSize = true;
button2.Text = "Previous";
button2.Location = new Point(170, 280);
button2.Click += Button2_Click;
this.Controls.Add(button2);

/* button3 */
button3 = new Button();
button3.AutoSize = true;
button3.Text = "Next";
button3.Location = new Point(350, 280);
button3.Click += Button3_Click;
this.Controls.Add(button3);

/* button4 */
button4 = new Button();
button4.AutoSize = true;
button4.Text = "Last";
button4.Location = new Point(475, 280);
button4.Click += Button4_Click;
this.Controls.Add(button4);

}

private void Form1_Load(object sender, EventArgs e)
{
```

```

/* create reference variables */
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
DataSet ds;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];

//display record
textbox1.Text = "0";
int n = Convert.ToInt32(textbox1.Text);
drow = dt.Rows[n];
textbox2.Text = Convert.ToString(drow["EmpID"]);
textbox3.Text = Convert.ToString(drow["EmpName"]);
textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button1_Click(object sender, EventArgs e)
{
    //first
    int n = 0;
    textbox1.Text = Convert.ToString(n);
}

```

```
drow = dt.Rows[n];
textbox2.Text = Convert.ToString(drow["EmpID"]);
textbox3.Text = Convert.ToString(drow["EmpName"]);
textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button2_Click(object sender, EventArgs e)
{
    //previous
    int n = Convert.ToInt32(textBox1.Text);
    n--;
    if (n == -1)
    {
        n = 0;
        MessageBox.Show("Already at first record");
    }
    textBox1.Text = Convert.ToString(n);
    drow = dt.Rows[n];
    textbox2.Text = Convert.ToString(drow["EmpID"]);
    textbox3.Text = Convert.ToString(drow["EmpName"]);
    textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button3_Click(object sender, EventArgs e)
{
    //next
    int n = Convert.ToInt32(textBox1.Text);
    n++;
    if (n == dt.Rows.Count)
    {
        n = dt.Rows.Count - 1;
        MessageBox.Show("Already at last record");
    }
    textBox1.Text = Convert.ToString(n);
    drow = dt.Rows[n];
    textbox2.Text = Convert.ToString(drow["EmpID"]);
    textbox3.Text = Convert.ToString(drow["EmpName"]);
```

```

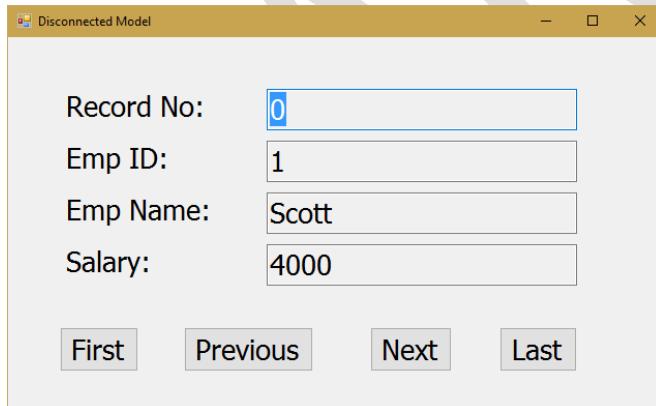
    textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button4_Click(object sender, EventArgs e)
{
    //last
    int n = dt.Rows.Count - 1;
    textbox1.Text = Convert.ToString(n);
    drow = dt.Rows[n];
    textbox2.Text = Convert.ToString(drow["EmpID"]);
    textbox3.Text = Convert.ToString(drow["EmpName"]);
    textbox4.Text = Convert.ToString(drow["Salary"]);
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

First record appears automatically. Click on “First”, “Previous”, “Next” and “Last” buttons.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – ComboBox - Example**Creating Database**

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ComboBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ComboBoxExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```

using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace ComboBoxExample
{
    public partial class Form1 : Form
    {
        DataTable dt;
        DataRow drow;
        Label label1, label2, label3;
        TextBox textbox1, textbox2;
        ComboBox combobox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
        }
    }
}

```

```
this.Text = "Disconnected Model";
this.StartPosition = FormStartPosition.CenterScreen;
this.Load += Form1_Load;

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Emp ID: ";
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary: ";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* combobox1 */
combobox1 = new ComboBox();
combobox1.Size = new Size(300, 50);
combobox1.DropDownStyle = ComboBoxStyle.DropDownList;
combobox1.Location = new Point(250, 50);
combobox1.SelectedIndexChanged +=
    Combbox1_SelectedIndexChanged;
this.Controls.Add(combobox1);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 100);
```

```
textbox1.ReadOnly = true;
textbox1.Location = new Point(250, 100);
this.Controls.Add(textBox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 150);
textbox2.ReadOnly = true;
textbox2.Location = new Point(250, 150);
this.Controls.Add(textBox2);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;

    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];
```

```

/* load empid's into combobox1 */
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    combobox1.Items.Add(drow["EmpID"]);
}

private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    int currentempid = Convert.ToInt32(combobox1.SelectedItem);

    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlParameter p1;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    p1 = new SqlParameter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select empname, salary from Employees where
empid=@empid";
    cmd.Connection = cn;
    p1.ParameterName = "@empid";
    p1.Value = currentempid;
    adp.SelectCommand = cmd;
    cmd.Parameters.Add(p1);
}

```

```

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];

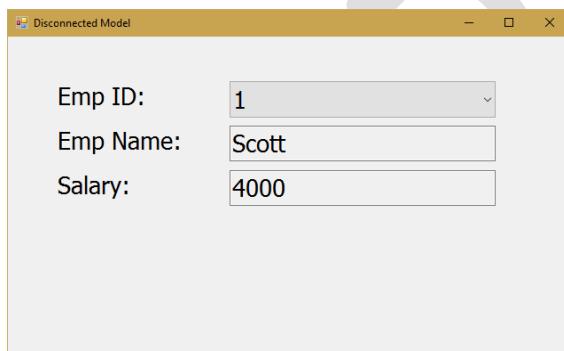
/* display selected emp details in textboxes */
drow = dt.Rows[0];
textbox1.Text = Convert.ToString(drow["empname"]);
textbox2.Text = Convert.ToString(drow["salary"]);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



All the EmpID's appear in the ComboBox. When the user selects an item in the ComboBox, it shows corresponding EmpName and Salary.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – ComboBox - DataSource - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DataSourceExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSourceExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DataSourceExample
{
    public partial class Form1 : Form
    {
        DataTable dt;
        DataRow drow;
        Label label1, label2, label3;
        TextBox textbox1, textbox2;
        ComboBox combobox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "Disconnected Model";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Emp ID: ";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* label2 */
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            string query = "SELECT * FROM Employees";
            SqlConnection connection = new SqlConnection("Data Source=.;Initial Catalog=Northwind;Integrated Security=True");
            SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
            adapter.Fill(dt);
            drow = dt.NewRow();
            drow["EmployeeID"] = 1;
            drow["FirstName"] = "John";
            drow["LastName"] = "Doe";
            drow["Title"] = "Software Developer";
            drow["HireDate"] = DateTime.Parse("1996-07-01");
            drow["Address"] = "123 Main St";
            drow["City"] = "Redmond";
            drow["Region"] = "WA";
            drow["PostalCode"] = "98052";
            drow["Country"] = "USA";
            drow["Phone"] = "(425) 555-0101";
            drow["Fax"] = "(425) 555-0102";
            drow["Email"] = "J.Doe@Microsoft.com";
            dt.Rows.Add(drow);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (label1.Text == "Emp ID: ")
            {
                label1.Text = "Employee ID: " + drow["EmployeeID"];
            }
            else
            {
                label1.Text = "Employee ID: " + drow["EmployeeID"];
            }
        }
    }
}
```

```
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name:";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary:";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* combobox1 */
comboBox1 = new ComboBox();
comboBox1.Size = new Size(300, 50);
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox1.Location = new Point(250, 50);
comboBox1.SelectedIndexChanged +=
Combobox1_SelectedIndexChanged;
this.Controls.Add(comboBox1);

/* textbox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(300, 100);
textBox1.ReadOnly = true;
textBox1.Location = new Point(250, 100);
this.Controls.Add(textBox1);

/* textbox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(300, 150);
textBox2.ReadOnly = true;
textBox2.Location = new Point(250, 150);
this.Controls.Add(textBox2);
}
```

```

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;

    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];

    /*bind data to combobox1*/
    combobox1.DataSource = dt;
    combobox1.DisplayMember = "EmpID";
}

private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    object si = combobox1.SelectedItem;
    DataRowView view = (DataRowView)si;
    object eid = view["EmpID"];
    int currentempid = Convert.ToInt32(eid);
}

```

```
/* create reference variables */
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
SqlParameter p1;
DataSet ds;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
p1 = new SqlParameter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
cmd.CommandText = "select empname, salary from Employees where empid=@empid";
cmd.Connection = cn;
p1.ParameterName = "@empid";
p1.Value = currentempid;
adp.SelectCommand = cmd;

/* call methods */
cmd.Parameters.Add(p1);
adp.Fill(ds);
dt = ds.Tables[0];

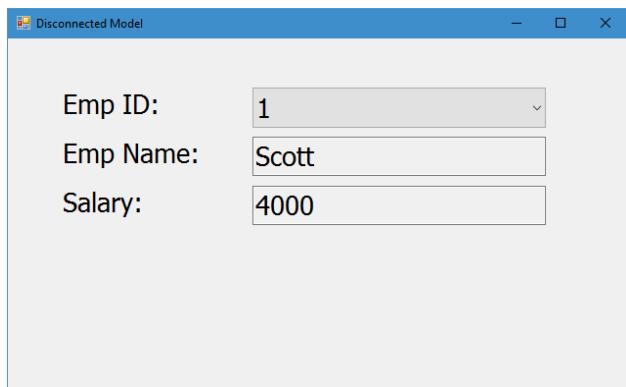
/* display selected emp details in textboxes */
drow = dt.Rows[0];
textBox1.Text = Convert.ToString(drow["empname"]);
textBox2.Text = Convert.ToString(drow["salary"]);
}

}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

All the EmpID's appear in the ComboBox. When the user selects an item in the ComboBox, it shows corresponding EmpName and Salary.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – DataGridView - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DataGridViewExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “DataGridViewExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DataGridViewExample
{
    public partial class Form1 : Form
    {
        DataGridView datagridview1;
        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "Disconnected Model";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;
            /* datagridview1 */
            datagridview1 = new DataGridView();
            datagridview1.Size = new Size(500, 300);
            datagridview1.Location = new Point(50, 50);
            datagridview1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            datagridview1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            this.Controls.Add(datagridview1);
        }
        private void Form1_Load(object sender, EventArgs e)
        {
```

```

/* create reference variables */
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
DataSet ds;
DataTable dt;
/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();
/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];

/* load employees table into datagridview1 */
datagridview1.DataSource = dt;
datagridview1.AutoResizeColumnHeadersHeight();
}

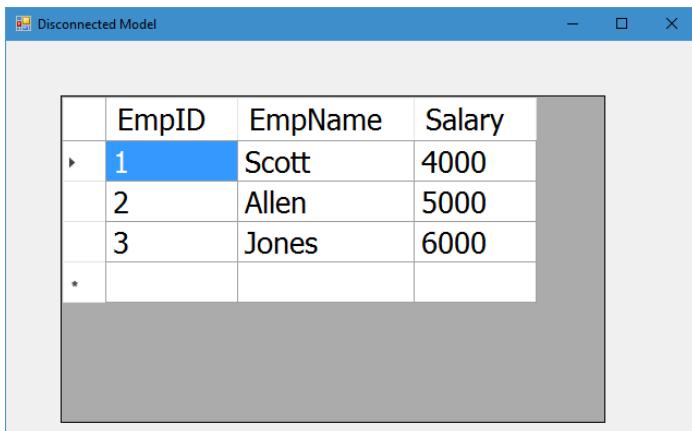
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



All the employees data appear in the DataGridView.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – DataGridView - SqlCommandBuilder - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”. Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DataGridViewExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataGridViewExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace SqlCommandBuilderExample
{
    public partial class Form1 : Form
    {
        SqlDataAdapter adp;
```

```
DataTable dt;
DataGridView datagridview1;
Button button1;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(650, 400);
    this.Text = "Disconnected Model";
    this.StartPosition = FormStartPosition.CenterScreen;
    this.Load += Form1_Load;

    /* datagridview1 */
    datagridview1 = new DataGridView();
    datagridview1.Size = new Size(500, 200);
    datagridview1.Location = new Point(50, 50);
    datagridview1.AutoSizeRowsMode =
        DataGridViewAutoSizeRowsMode.AllCells;
    datagridview1.AutoSizeColumnsMode =
        DataGridViewAutoSizeColumnsMode.AllCells;
    this.Controls.Add(datagridview1);

    /* button1 */
    button1 = new Button();
    button1.AutoSize = true;
    button1.Text = "Save";
    button1.Location = new Point(470, 280);
    button1.Click += Button1_Click;
    this.Controls.Add(button1);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
```

```
SqlConnection cn;
SqlCommand cmd;
SqlCommandBuilder cmb;
DataSet ds;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();
cmb = new SqlCommandBuilder();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;
cmbDataAdapter = adp;

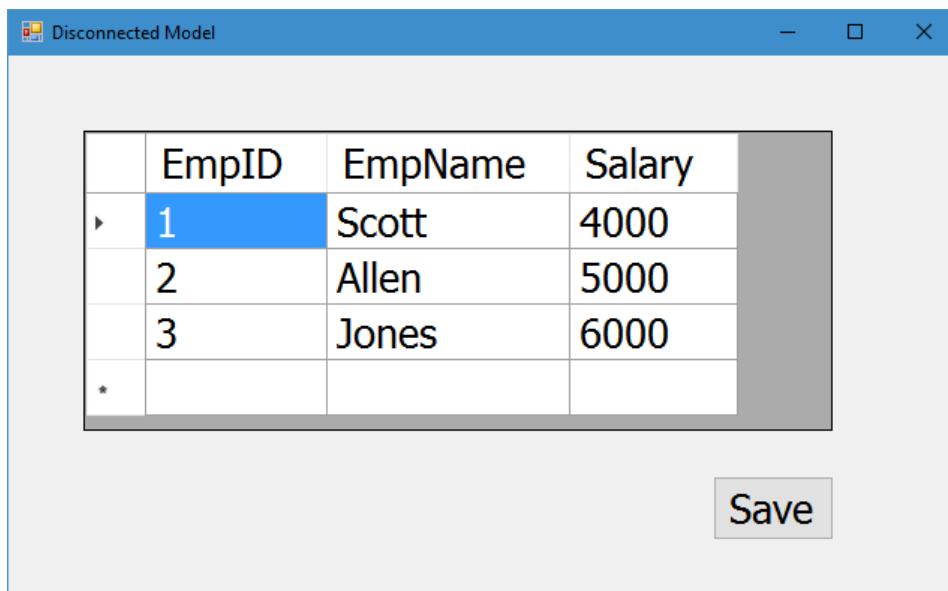
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];

/* load employees table into datagridview1 */
datagridview1.DataSource = dt;
datagridview1.AutoResizeColumnHeadersHeight();
}

private void Button1_Click(object sender, EventArgs e)
{
    adp.Update(dt);
    MessageBox.Show("Saved");
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

All the employees data appear in the DataGridView.

Make some insertion, deletion and updation in the DataGridView, and click on “Save” button. The changes will be saved in the database.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Master – Child – Example

Creating Database

- **Note:** Ignore this step, if you have created “departmentsandemployeesdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database departmentsandemployeesdatabase  
go
```

```
use departmentsandemployeesdatabase  
go
```

```
create table Departments(  
DeptNo int primary key,  
DeptName nvarchar(max),  
Loc nvarchar(max))  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName varchar(max),  
Salary decimal,  
DeptNo int references Departments(DeptNo))  
go
```

```
insert into Departments values(10, 'Acounting', 'New York')  
insert into Departments values(20, 'Operations', 'New Delhi')  
insert into Departments values(30, 'Sales', 'New Jersy')
```

```
insert into Employees values(1, 'Scott', 3000, 10)  
insert into Employees values(2, 'Allen', 6500, 10)  
insert into Employees values(3, 'Jones', 4577, 20)  
insert into Employees values(4, 'James', 9500, 20)  
insert into Employees values(5, 'Smith', 3345, 30)  
insert into Employees values(6, 'Harry', 2500, 30)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MasterChildExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MasterChildExample”. Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace MasterChildExample
{
    public partial class Form1 : Form
    {
        DataTable dt;
        DataGridView datagridview1;
        Label label1;
        ComboBox combobox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(750, 400);
            this.Text = "Disconnected Model";
            this.StartPosition = FormStartPosition.CenterScreen;
        }
    }
}
```

```

this.Load += Form1_Load;

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Dept No: ";
label1.Location = new Point(105, 52);
this.Controls.Add(label1);

/* combobox1 */
comboBox1 = new ComboBox();
comboBox1.Size = new Size(300, 50);
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox1.Location = new Point(250, 50);
comboBox1.SelectedIndexChanged +=
    Combobox1_SelectedIndexChanged;
this.Controls.Add(comboBox1);

/* datagridview1 */
dataGridView1 = new DataGridView();
dataGridView1.Size = new Size(600, 200);
dataGridView1.Location = new Point(50, 120);
dataGridView1.AutoSizeRowsMode =
    DataGridViewAutoSizeRowsMode.AllCells;
dataGridView1.AutoSizeColumnsMode =
    DataGridViewAutoSizeColumnsMode.AllCells;
this.Controls.Add(dataGridView1);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;
}

```

```

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=departmentsandemployeesdatabase";
cmd.CommandText = "select * from Departments";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];

/*bind data to comboBox1*/
combobox1.DataSource = dt;
combobox1.DisplayMember = "DeptName";
combobox1.ValueMember = "DeptNo";
}

private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlParameter p1;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
}

```

```
p1 = new SqlParameter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=departmentsandemployeesdatabase";
cmd.CommandText = "select * from Employees where
DeptNo=@DeptNo";
cmd.Connection = cn;
adp.SelectCommand = cmd;
p1.ParameterName = "@DeptNo";
p1.Value = ((DataRowView)combobox1.SelectedItem)[("DeptNo")];
cmd.Parameters.Add(p1);

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];

/* load employees table into datagridview1 */
datagridview1.DataSource = dt;
datagridview1.AutoResizeColumnHeadersHeight();

}

}

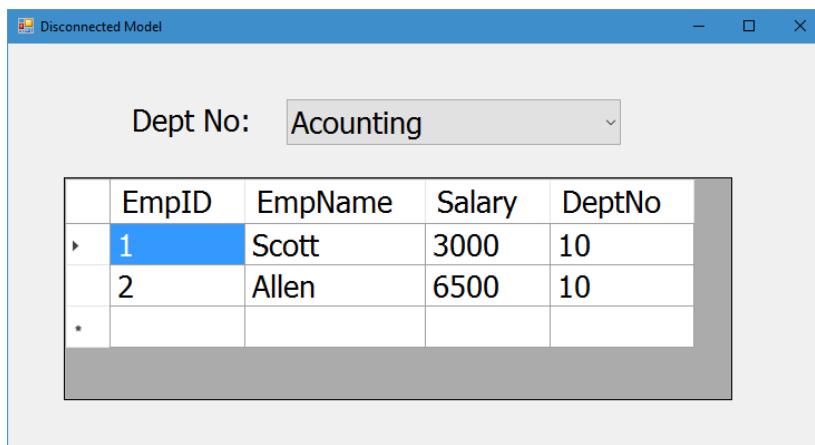
}

}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows all the department names in the ComboBox.

When the user selects a department name, it shows all the corresponding employees in the DataGridView.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Cascading ComboBox – Example

Creating Database

- Note: Ignore this step, if you have created “CountriesAndStatesDatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database CountriesAndStatesDatabase
go
```

```
use CountriesAndStatesDatabase
go
```

```
create table Countries(
CountryID int primary key,
```

```

CountryName nvarchar(max))
go

create table States(
StateID int primary key,
StateName varchar(max),
CountryID int references Countries(CountryID))
go

insert into Countries values(1, 'India')
insert into Countries values(2, 'USA')
insert into Countries values(3, 'UK')

insert into States values(1, 'Tamilnadu', 1)
insert into States values(2, 'Maharashtra', 1)
insert into States values(3, 'Karnataka', 1)
insert into States values(4, 'Kerala', 1)

insert into States values(5, 'Washington', 2)
insert into States values(6, 'New Jersey', 2)
insert into States values(7, 'New York', 2)
insert into States values(8, 'Texas', 2)

insert into States values(9, 'Moray', 3)
insert into States values(10, 'Norfolk', 3)
insert into States values(11, 'Bolton', 3)
insert into States values(12, 'Newport', 3)
go

```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “CascadingComboBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CascadingComboBoxExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace CascadingComboBoxExample
{
    public partial class Form1 : Form
    {
        Label label1, label2;
        ComboBox combobox1, combobox2;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(750, 400);
            this.Text = "Disconnected Model";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;
        }
    }
}
```

```

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Country: ";
label1.Location = new Point(105, 52);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "State: ";
label2.Location = new Point(105, 132);
this.Controls.Add(label2);

/* combobox1 */
comboBox1 = new ComboBox();
comboBox1.Size = new Size(300, 50);
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox1.Location = new Point(250, 50);
comboBox1.SelectedIndexChanged +=
    Combobox1_SelectedIndexChanged;
this.Controls.Add(comboBox1);

/* combobox2 */
comboBox2 = new ComboBox();
comboBox2.Size = new Size(300, 50);
comboBox2.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox2.Location = new Point(250, 130);
this.Controls.Add(comboBox2);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
}

```

```

DataSet ds;
DataTable dt;
/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();
/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=CountriesAndStatesDatabase";
cmd.CommandText = "select * from Countries";
cmd.Connection = cn;
adp.SelectCommand = cmd;
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
/*bind data to combobox1*/
combobox1.DataSource = dt;
combobox1.DisplayMember = "CountryName";
combobox1.ValueMember = "CountryID";
}
private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
/* create reference variables */
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
SqlParameter p1;
DataSet ds;
DataTable dt;
/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
p1 = new SqlParameter();
ds = new DataSet();

```

```

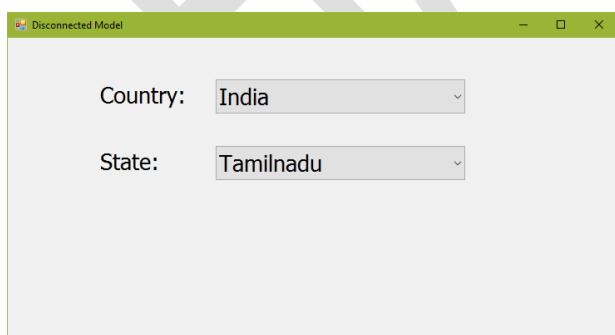
/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=CountriesAndStatesDatabase";
cmd.CommandText = "select * from States where
CountryID=@CountryID";
cmd.Connection = cn;
adp.SelectCommand = cmd;
p1.ParameterName = "@CountryID";
p1.Value = ((DataRowView)comboBox1.SelectedItem)[("CountryID")];
cmd.Parameters.Add(p1);
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
/*bind data to comboBox2*/
comboBox2.DataSource = dt;
comboBox2.DisplayMember = "StateName";
comboBox2.ValueMember = "StateID";
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



It shows all the country names in the first ComboBox.

When the user selects a country name, it shows all the corresponding states in the second ComboBox.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query - Insertion – Example

Creating Database

- **Note:** Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “InsertionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InsertionExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace InsertionExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "ADO.NET NonQuery";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Insert";
        }
    }
}
```

```

button1.AutoSize = true;
button1.Location = new Point(170, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);
}

private void button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "insert into Employees values(10, 'qwerty',
4500)";
    cmd.Connection = cn;
    //calling methods
    cn.Open();
    int n = cmd.ExecuteNonQuery();
    cn.Close();

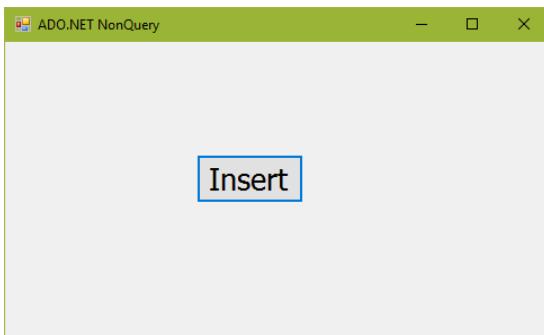
    string msg = n + " rows inserted";
    MessageBox.Show(msg);
}
}
}

```

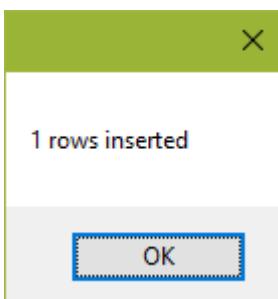
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Insert”.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query - Updation – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
EmpID int primary key,
```

```
EmpName nvarchar(max),  
Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “UpdationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationExample”. Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;  
using System.Windows.Forms;  
using System.Drawing;  
using System.Data.SqlClient;  
  
namespace UpdationExample  
{  
    public partial class Form1 : Form  
    {  
        Button button1;
```

```
public Form1()
{
    InitializeComponent();
    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(500, 300);
    this.Text = "ADO.NET NonQuery";
    this.StartPosition = FormStartPosition.CenterScreen;
    /* button1 */
    button1 = new Button();
    button1.Text = "Update";
    button1.AutoSize = true;
    button1.Location = new Point(170, 100);
    button1.Click += button1_Click;
    this.Controls.Add(button1);
}

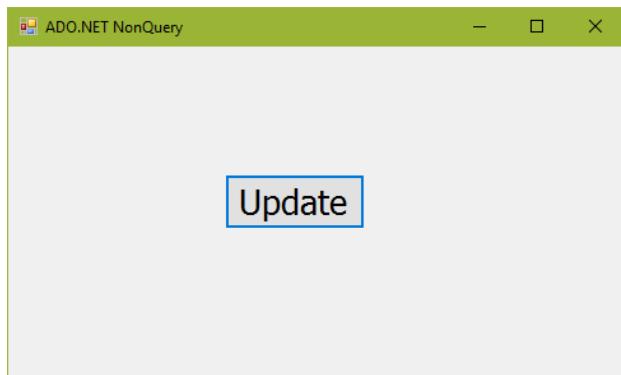
private void button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "update Employees set EmpName='asdf',
Salary=8900 where EmpID=1";
    cmd.Connection = cn;
    //calling methods
    cn.Open();
    int n = cmd.ExecuteNonQuery();
    cn.Close();
}
```

```
        string msg = n + " rows updated";
        MessageBox.Show(msg);
    }
}
}
```

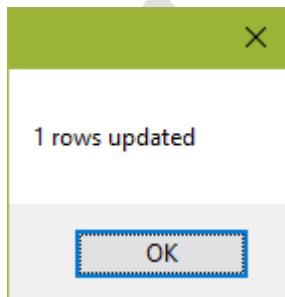
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Update”.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query - Deletion – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DeletionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DeletionExample”. Click on OK.

- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace DeletionExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "ADO.NET NonQuery";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Delete";
            button1.AutoSize = true;
            button1.Location = new Point(170, 100);
            button1.Click += button1_Click;
            this.Controls.Add(button1);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //create reference variables
```

```


SqlConnection cn;
SqlCommand cmd;

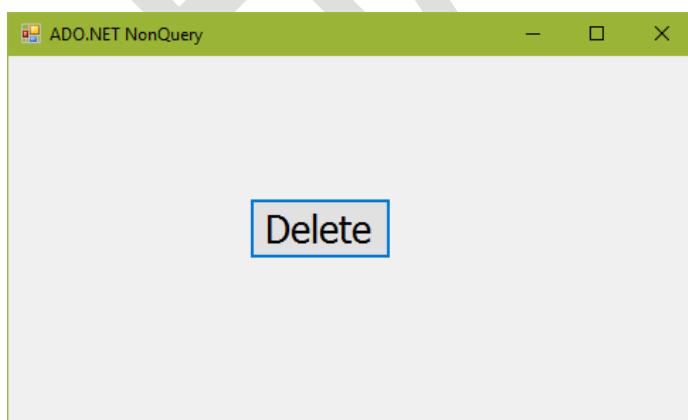
//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
//calling properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "delete from Employees where EmpID=3";
cmd.Connection = cn;
//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();
string msg = n + " rows deleted";
MessageBox.Show(msg);
}
}
}
}


```

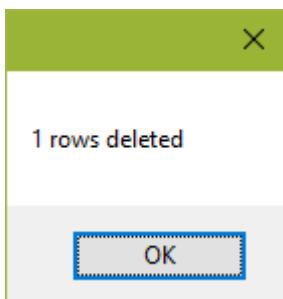
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “Delete”.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Insertion – With SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go

create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go

insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “InsertionSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “InsertionSqlParameterExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace InsertionSqlParameterExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1;

        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
/* form properties */
this.Font = new Font("Tahoma", 20);
this.Size = new Size(650, 400);
this.Text = "ADO.NET NonQuery - Insertion";
this.StartPosition = FormStartPosition.CenterScreen;

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Emp ID: ";
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary: ";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(250, 50);
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
```

```
textbox2.Location = new Point(250, 100);
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(250, 150);
this.Controls.Add(textbox3);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Insert";
button1.Location = new Point(250, 200);
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);

}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();

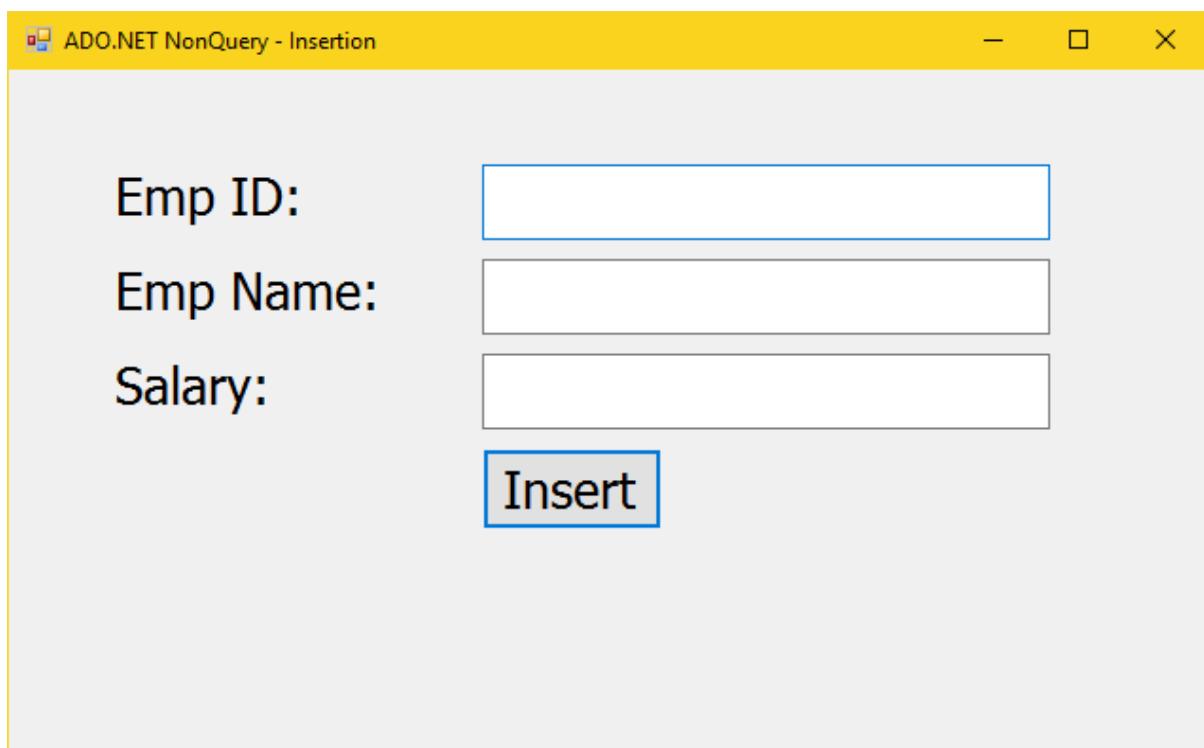
    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
```

```
cmd.CommandText = "insert into Employees values(@empid,  
@empname, @salary);  
cmd.Connection = cn;  
p1.ParameterName = "@empid";  
p2.ParameterName = "@empname";  
p3.ParameterName = "@salary";  
p1.Value = textbox1.Text;  
p2.Value = textbox2.Text;  
p3.Value = textbox3.Text;  
cmd.Parameters.Add(p1);  
cmd.Parameters.Add(p2);  
cmd.Parameters.Add(p3);  
  
//calling methods  
cn.Open();  
int n = cmd.ExecuteNonQuery();  
cn.Close();  
  
string msg = n + " rows inserted";  
MessageBox.Show(msg);  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Type some employee details and click on “Insert”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Updation – With SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “UpdationSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationSqlParameterExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
```

```
namespace UpdationSqlParameterExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "ADO.NET NonQuery - Updation";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Existing Emp ID: ";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* label2 */
            label2 = new Label();
            label2.AutoSize = true;
            label2.Text = "Emp Name: ";
            label2.Location = new Point(50, 100);
            this.Controls.Add(label2);

            /* label3 */
            label3 = new Label();
            label3.AutoSize = true;
            label3.Text = "Salary:";
            label3.Location = new Point(50, 150);
        }
    }
}
```

```
this.Controls.Add(label3);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(300, 50);
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(300, 100);
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(300, 150);
this.Controls.Add(textbox3);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Update";
button1.Location = new Point(300, 200);
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);

}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;
```

```

//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
p1 = new SqlParameter();
p2 = new SqlParameter();
p3 = new SqlParameter();

//calling properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "update Employees set empname=@empname,
salary=@salary where empid=@empid";
cmd.Connection = cn;
p1.ParameterName = "@empid";
p2.ParameterName = "@empname";
p3.ParameterName = "@salary";
p1.Value = textbox1.Text;
p2.Value = textbox2.Text;
p3.Value = textbox3.Text;
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);

//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

string msg = n + " rows updated";
MessageBox.Show(msg);
}

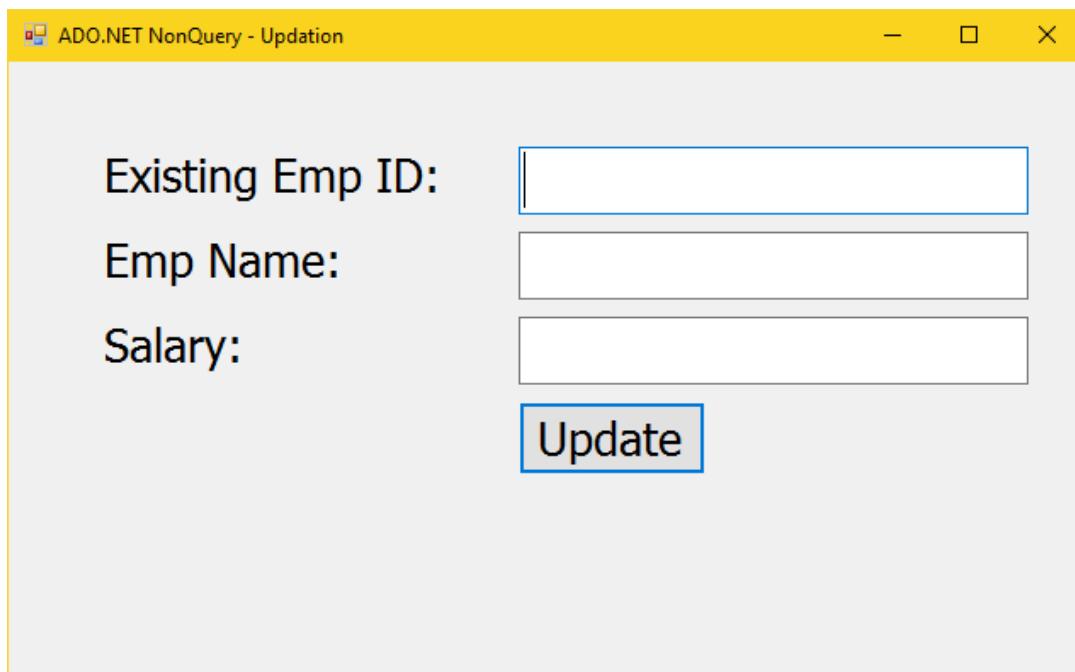
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some existing emp id, new emp name, new salary and click on “Update”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Deletion – With SqlParameter – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(  
    EmpID int primary key,  
    EmpName nvarchar(max),  
    Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DeletionSqlParameterExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DeletionSqlParameterExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;  
using System.Windows.Forms;  
using System.Drawing;  
using System.Data.SqlClient;
```

```
namespace DeletionSqlParameterExample
{
    public partial class Form1 : Form
    {
        Label label1;
        TextBox textbox1;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 300);
            this.Text = "ADO.NET NonQuery - Deletion";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Existing Emp ID: ";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);

            /* textbox1 */
            textbox1 = new TextBox();
            textbox1.Size = new Size(200, 50);
            textbox1.Location = new Point(300, 50);
            this.Controls.Add(textbox1);

            /* button1 */
            button1 = new Button();
            button1.AutoSize = true;
            button1.Text = "Delete";
            button1.Location = new Point(300, 120);
            this.AcceptButton = button1;
        }
    }
}
```

```

button1.Click += Button1_Click;
this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "delete from Employees where empid=@empid";
    cmd.Connection = cn;
    p1.ParameterName = "@empid";
    p1.Value = textBox1.Text;
    cmd.Parameters.Add(p1);

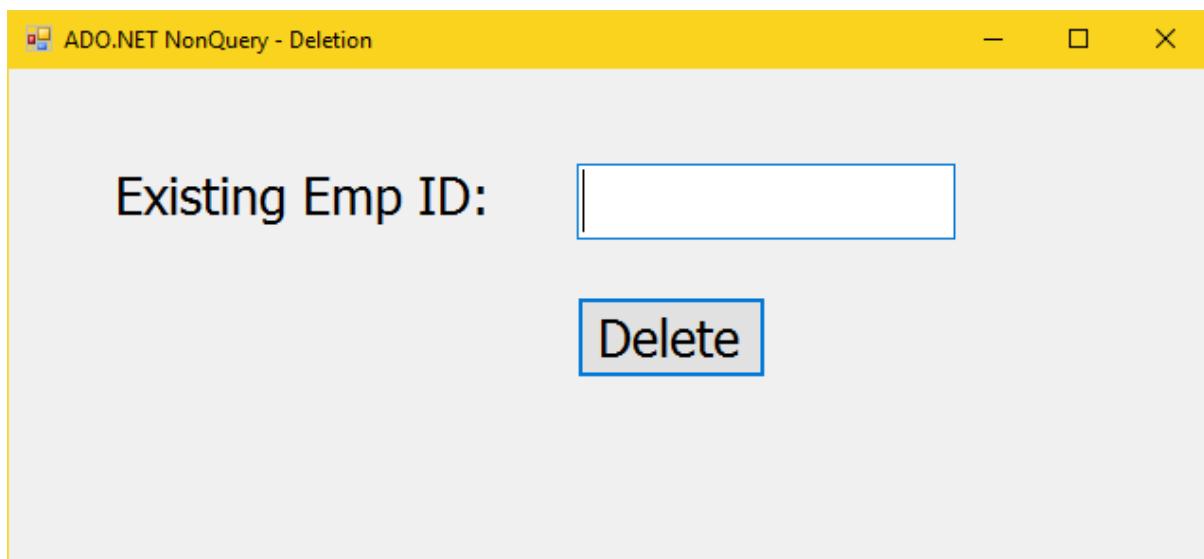
    //calling methods
    cn.Open();
    int n = cmd.ExecuteNonQuery();
    cn.Close();

    string msg = n + " rows deleted";
    MessageBox.Show(msg);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Enter some existing employee ID and click on “Delete”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Insertion – With Stored Procedures – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database company

go

```
use company
```

```
go
```

```
create table Employees(  

EmpID int primary key,  

EmpName nvarchar(max),  

Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  

insert into Employees values(2, 'Allen', 5000)  

insert into Employees values(3, 'Jones', 6000)  

go
```

```
create procedure InsertEmployee(  

@EmpID int,  

@EmpName nvarchar(max),  

@Salary decimal  

<>)  

as begin  

    insert into Employees values(@EmpID, @EmpName, @Salary)  

end  

go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “InsertionWithSPExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “InsertionWithSPExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace InsertionWithSPExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "ADO.NET NonQuery - Insertion - Stored Procedure";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Emp ID: ";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);
```

```
/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary: ";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(250, 50);
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(250, 100);
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(250, 150);
this.Controls.Add(textbox3);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Insert";
button1.Location = new Point(250, 200);
```

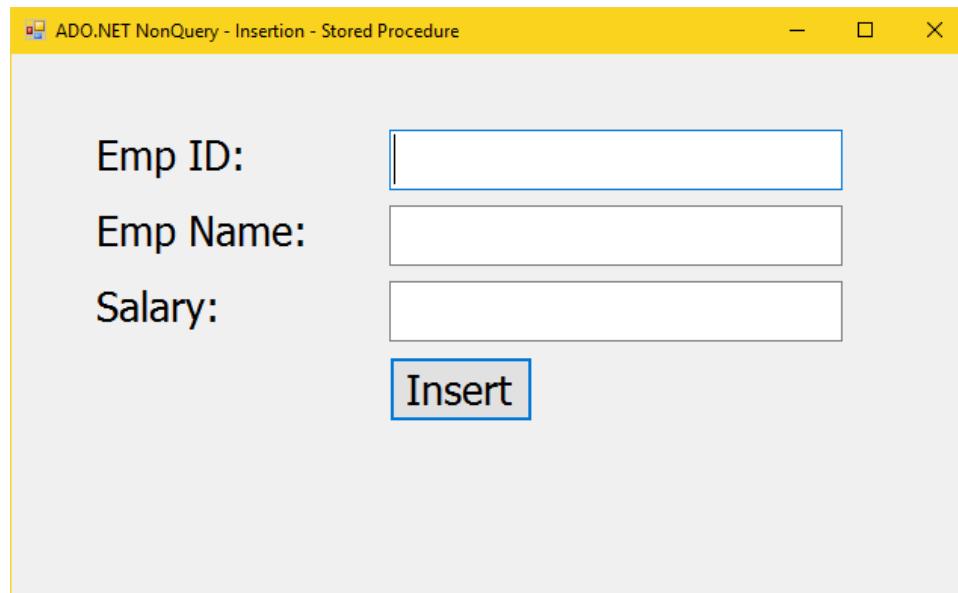
```
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);
}
private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;
    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();
    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "InsertEmployee";
    cmd.Connection = cn;
    cmd.CommandType = CommandType.StoredProcedure;
    p1.ParameterName = "@empid";
    p2.ParameterName = "@empname";
    p3.ParameterName = "@salary";
    p1.Value = textBox1.Text;
    p2.Value = textBox2.Text;
    p3.Value = textBox3.Text;
    cmd.Parameters.Add(p1);
    cmd.Parameters.Add(p2);
    cmd.Parameters.Add(p3);
    //calling methods
    cn.Open();
    int n = cmd.ExecuteNonQuery();
    cn.Close();
    MessageBox.Show(n + " rows inserted");
}
```

```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Enter some emp id, ename, salary and click on “Insert”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Updation – With Stored Procedures – Example**Creating Database**

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

```
create procedure UpdateEmployee(
```

```
    @EmpID int,
```

```
    @EmpName nvarchar(max),
```

```
    @Salary decimal
```

```
)
```

```
as begin
```

```
    update Employees set EmpName=@EmpName, Salary=@Salary
```

```
    where EmpID=@EmpID
```

```
end
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “UpdationWithSPExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationWithSPExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace UpdationWithSPExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "ADO.NET NonQuery - Updation - Stored Procedure";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
        }
    }
}
```

```
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Existing Emp ID: ";
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary:";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(300, 50);
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(300, 100);
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(300, 150);
this.Controls.Add(textbox3);
```

```
/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Update";
button1.Location = new Point(300, 200);
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "UpdateEmployee";
    cmd.Connection = cn;
    cmd.CommandType = CommandType.StoredProcedure;
    p1.ParameterName = "@empid";
    p2.ParameterName = "@empname";
    p3.ParameterName = "@salary";
    p1.Value = textbox1.Text;
    p2.Value = textbox2.Text;
    p3.Value = textbox3.Text;
```

```
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);

//calling methods
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

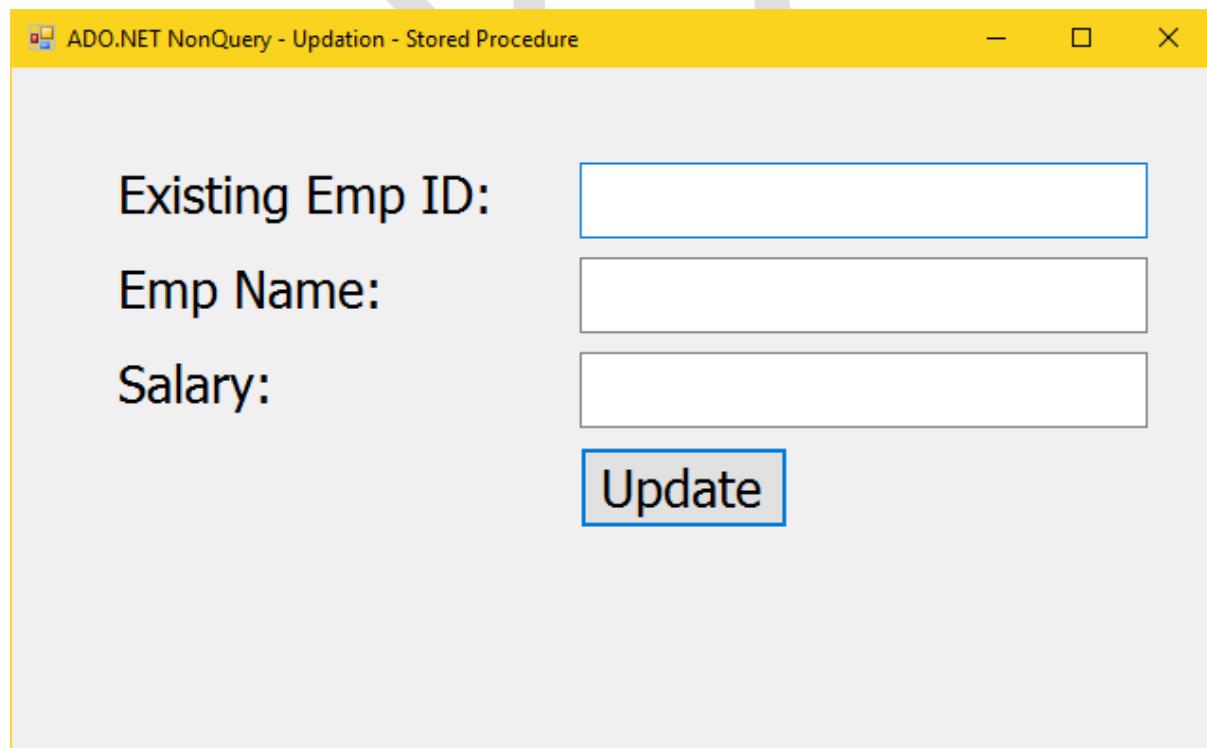
MessageBox.Show(n + " rows updated");
}

}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some existing emp id, new emp name, new salary and click on “Update”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Non Query – Deletion – With Stored Procedures – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company  
go
```

```
use company  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

```
create procedure DeleteEmployee(  
@EmpID int  
)  
as begin
```

```
delete from Employees
where EmpID=@EmpID
end
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “DeletionWithSPExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DeletionWithSPExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DeletionWithSPExample
{
    public partial class Form1 : Form
    {
        Label label1;
```

```
TextBox textbox1;
Button button1;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(650, 300);
    this.Text = "ADO.NET NonQuery - Deletion - Stored Procedure";
    this.StartPosition = FormStartPosition.CenterScreen;

    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = "Existing Emp ID: ";
    label1.Location = new Point(50, 50);
    this.Controls.Add(label1);

    /* textbox1 */
    textbox1 = new TextBox();
    textbox1.Size = new Size(300, 50);
    textbox1.Location = new Point(300, 50);
    this.Controls.Add(textbox1);

    /* button1 */
    button1 = new Button();
    button1.AutoSize = true;
    button1.Text = "Delete";
    button1.Location = new Point(300, 100);
    this.AcceptButton = button1;
    button1.Click += Button1_Click;
    this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
```

```

{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "DeleteEmployee";
    cmd.Connection = cn;
    cmd.CommandType = CommandType.StoredProcedure;
    p1.ParameterName = "@empid";
    p1.Value = textBox1.Text;
    cmd.Parameters.Add(p1);

    //calling methods
    cn.Open();
    int n = cmd.ExecuteNonQuery();
    cn.Close();

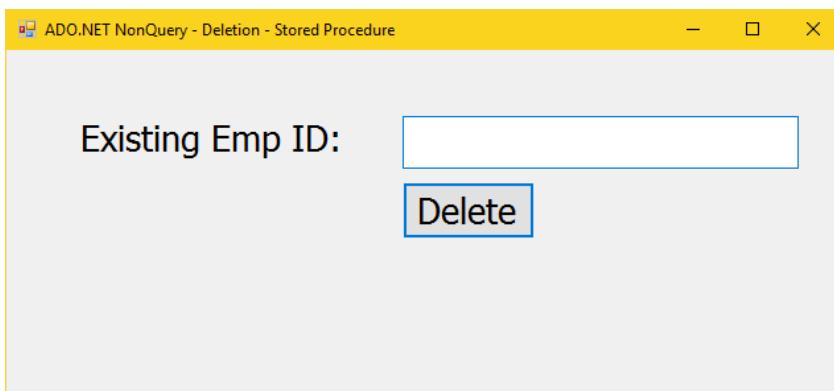
    MessageBox.Show(n + " rows deleted");
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some existing emp id and click on “Delete”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – ComboBox – Stored Procedures – Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
    EmpID int primary key,
    EmpName nvarchar(40),
    Salary decimal
```

```
)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6500)
insert into Employees values(4, 'James', 7000)
go
```

```
create procedure GetEmployees
as begin
    select * from Employees
end
go
```

```
create procedure GetEmployeeByID(
    @EmpID int
)
as begin
    select * from Employees where EmpID=@EmpID
end
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ComboBoxWithSPEexample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “ComboBoxWithSPExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace ComboBoxWithSPExample
{
    public partial class Form1 : Form
    {
        DataTable dt;
        DataRow draw;
        Label label1, label2, label3;
        TextBox textbox1, textbox2;
        ComboBox combobox1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "ADO.NET Stored Procedures - ComboBox";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Emp ID: ";
        }
    }
}
```

```
label1.Location = new Point(50, 50);
this.Controls.Add(label1);
```

```
/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);
```

```
/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary:";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);
```

```
/* combobox1 */
comboBox1 = new ComboBox();
comboBox1.Size = new Size(300, 50);
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox1.Location = new Point(250, 50);
comboBox1.SelectedIndexChanged +=
    Combobox1_SelectedIndexChanged;
this.Controls.Add(comboBox1);
```

```
/* textbox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(300, 100);
textBox1.ReadOnly = true;
textBox1.Location = new Point(250, 100);
this.Controls.Add(textBox1);
```

```
/* textbox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(300, 150);
textBox2.ReadOnly = true;
```

```
textbox2.Location = new Point(250, 150);
this.Controls.Add(textbox2);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "GetEmployees";
    cmd.Connection = cn;
    cmd.CommandType = CommandType.StoredProcedure;
    adp.SelectCommand = cmd;

    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];

    /* load empid's into combobox1 */
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        drow = dt.Rows[i];
        combobox1.Items.Add(drow["EmpID"]);
    }
}
```

```
private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    int currentempid = Convert.ToInt32(comboBox1.SelectedItem);

    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlParameter p1;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    p1 = new SqlParameter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "GetEmployeeByID";
    cmd.Connection = cn;
    cmd.CommandType = CommandType.StoredProcedure;
    p1.ParameterName = "@empid";
    p1.Value = currentempid;
    adp.SelectCommand = cmd;

    /* call methods */
    cmd.Parameters.Add(p1);
    adp.Fill(ds);
    dt = ds.Tables[0];

    /* display selected emp details in textboxes */
    drow = dt.Rows[0];
```

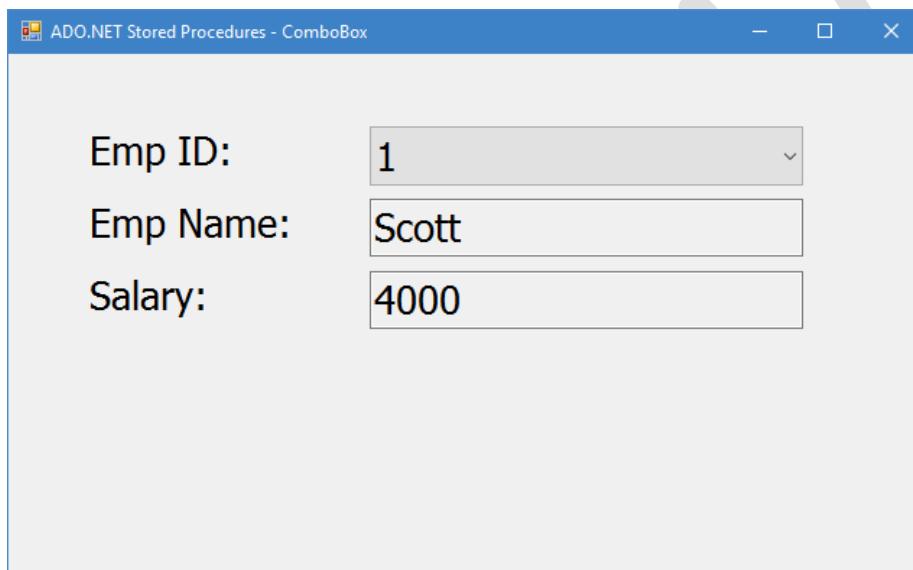
```

        textbox1.Text = Convert.ToString(drow["empname"]);
        textbox2.Text = Convert.ToString(drow["salary"]);
    }
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

All the existing EmpID's appear in the combobox. Select an EmpID, then it shows corresponding EmpName and Salary.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET Disconnected Model – Updation with ComboBox - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.

- Click on “New Query”. Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
    EmpID int primary key,
```

```
    EmpName nvarchar(40),
```

```
    Salary decimal
```

```
)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6500)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “UpdationWithComboBoxExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “UpdationWithComboBoxExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace UpdationWithComboBoxExample
{
    public partial class Form1 : Form
    {
        DataTable dt;
        DataRow drow;
        Label label1, label2, label3;
        TextBox textbox1, textbox2;
        ComboBox combobox1;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "Updation with ComboBox Example";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Emp ID: ";
            label1.Location = new Point(50, 50);
            this.Controls.Add(label1);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            string query = "SELECT * FROM Employees";
            SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
            adapter.Fill(dt);

            if (dt != null && dt.Rows.Count > 0)
            {
                drow = dt.NewRow();
                drow["EmployeeID"] = 1;
                drow["EmployeeName"] = "John Doe";
                drow["EmployeeAddress"] = "123 Main St, Anytown, USA";
                drow["EmployeeSalary"] = 50000;
                dt.Rows.Add(drow);
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (label1.Text == "Emp ID: ")
            {
                label1.Text = "Employee ID updated!";
            }
            else
            {
                label1.Text = "Employee ID updated!";
            }
        }
    }
}
```

```
/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary: ";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* combobox1 */
comboBox1 = new ComboBox();
comboBox1.Size = new Size(300, 50);
comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
comboBox1.Location = new Point(250, 50);
comboBox1.SelectedIndexChanged +=
    Combobox1_SelectedIndexChanged;
this.Controls.Add(comboBox1);

/* textbox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(300, 100);
textBox1.Location = new Point(250, 100);
this.Controls.Add(textBox1);

/* textbox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(300, 150);
textBox2.Location = new Point(250, 150);
this.Controls.Add(textBox2);

/* button1 */
button1 = new Button();
```

```
button1.AutoSize = true;
button1.Text = "Update";
button1.Location = new Point(250, 200);
button1.Click += Button1_Click;
this.Controls.Add(button1);
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;

    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];

    /* load empid's into combobox1 */
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        drow = dt.Rows[i];
        combobox1.Items.Add(drow["EmpID"]);
    }
}
```

```
        }

    }

    private void Combobox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    int currentempid = Convert.ToInt32(combobox1.SelectedItem);

    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlParameter p1;
    DataSet ds;

    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    p1 = new SqlParameter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
    cmd.CommandText = "select empname, salary from Employees where empid=@empid";
    cmd.Connection = cn;
    p1.ParameterName = "@empid";
    p1.Value = currentempid;
    adp.SelectCommand = cmd;

    /* call methods */
    cmd.Parameters.Add(p1);
    adp.Fill(ds);
    dt = ds.Tables[0];
```

```

/* display selected emp details in textboxes */
drow = dt.Rows[0];
textbox1.Text = Convert.ToString(drow["empname"]);
textbox2.Text = Convert.ToString(drow["salary"]);
}

private void Button1_Click(object sender, EventArgs e)
{
    int currentempid = Convert.ToInt32(comboBox1.SelectedItem);

    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=company";
    cmd.CommandText = "update employees set empname=@empname, salary=@salary where empid=@empid";
    cmd.Connection = cn;
    p1.ParameterName = "@empid";
    p2.ParameterName = "@empname";
    p3.ParameterName = "@salary";
    p1.Value = currentempid;
    p2.Value = textBox1.Text;
    p3.Value = textBox2.Text;

    //calling methods
    cmd.Parameters.Add(p1);
}

```

```

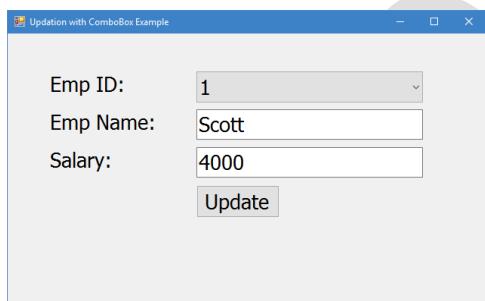
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();

string msg = n + " rows updated";
MessageBox.Show(msg);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

All the existing EmpID's appear in the combobox. Select an EmpID, then it shows corresponding EmpName and Salary.

Change the values of EmpName and Salary and then click on “Update”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – CRUD (Create, Retrieve, Update, Delete) - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”. Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
    EmpID int primary key,
```

```
    EmpName nvarchar(40),
```

```
    Salary decimal
```

```
)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6500)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “CRUDExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “CRUDExample”.
- Click on OK.

- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace CRUDExample
{
    public partial class Form1 : Form
    {
        DataTable dt;
        DataRow drow;
        Label label1, label2, label3, label4;
        TextBox textbox1, textbox2, textbox3, textbox4;
        Button button1, button2, button3, button4, button5, button6, button7,
button8;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 450);
            this.Text = "Disconnected Model";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Load += Form1_Load;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Record No:";
            label1.Location = new Point(50, 50);
```

```
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp ID: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Emp Name:";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* label4 */
label4 = new Label();
label4.AutoSize = true;
label4.Text = "Salary:";
label4.Location = new Point(50, 200);
this.Controls.Add(label4);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(250, 50);
textbox1.ReadOnly = true;
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(250, 100);
textbox2.ReadOnly = true;
this.Controls.Add(textbox2);
```

```
/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(250, 150);
this.Controls.Add(textbox3);

/* textbox4 */
textbox4 = new TextBox();
textbox4.Size = new Size(300, 50);
textbox4.Location = new Point(250, 200);
this.Controls.Add(textbox4);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "First";
button1.Location = new Point(50, 280);
button1.Click += Button1_Click;
this.Controls.Add(button1);

/* button2 */
button2 = new Button();
button2.AutoSize = true;
button2.Text = "Previous";
button2.Location = new Point(170, 280);
button2.Click += Button2_Click;
this.Controls.Add(button2);

/* button3 */
button3 = new Button();
button3.AutoSize = true;
button3.Text = "Next";
button3.Location = new Point(350, 280);
button3.Click += Button3_Click;
this.Controls.Add(button3);

/* button4 */
```

```
button4 = new Button();
button4.AutoSize = true;
button4.Text = "Last";
button4.Location = new Point(475, 280);
button4.Click += Button4_Click;
this.Controls.Add(button4);

/* button5 */
button5 = new Button();
button5.AutoSize = true;
button5.Text = "Create";
button5.Location = new Point(50, 340);
button5.Click += Button5_Click;
this.Controls.Add(button5);

/* button6 */
button6 = new Button();
button6.AutoSize = true;
button6.Text = "Insert";
button6.Location = new Point(170, 340);
button6.Visible = false;
button6.Click += Button6_Click;
this.Controls.Add(button6);

/* button7 */
button7 = new Button();
button7.AutoSize = true;
button7.Text = "Update";
button7.Location = new Point(350, 340);
button7.Click += Button7_Click;
this.Controls.Add(button7);

/* button8 */
button8 = new Button();
button8.AutoSize = true;
button8.Text = "Delete";
button8.Location = new Point(475, 340);
```

```
button8.Click += Button8_Click;
this.Controls.Add(button8);
}

private void loaddata()
{
/* create reference variables */
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
DataSet ds;

/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
}

private void Form1_Load(object sender, EventArgs e)
{
loaddata();

//display record
textbox1.Text = "0";
int n = Convert.ToInt32(textbox1.Text);
```

```
drow = dt.Rows[n];
textbox2.Text = Convert.ToString(drow["EmpID"]);
textbox3.Text = Convert.ToString(drow["EmpName"]);
textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button1_Click(object sender, EventArgs e)
{
    //first
    int n = 0;
    textbox1.Text = Convert.ToString(n);
    drow = dt.Rows[n];
    textbox2.Text = Convert.ToString(drow["EmpID"]);
    textbox3.Text = Convert.ToString(drow["EmpName"]);
    textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button2_Click(object sender, EventArgs e)
{
    //previous
    int n = Convert.ToInt32(textbox1.Text);
    n--;
    if (n == -1)
    {
        n = 0;
        MessageBox.Show("Already at first record");
    }
    textbox1.Text = Convert.ToString(n);
    drow = dt.Rows[n];
    textbox2.Text = Convert.ToString(drow["EmpID"]);
    textbox3.Text = Convert.ToString(drow["EmpName"]);
    textbox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button3_Click(object sender, EventArgs e)
{
    //next
}
```

```
int n = Convert.ToInt32(textBox1.Text);
n++;
if (n == dt.Rows.Count)
{
    n = dt.Rows.Count - 1;
    MessageBox.Show("Already at last record");
}
textBox1.Text = Convert.ToString(n);
drow = dt.Rows[n];
textBox2.Text = Convert.ToString(drow["EmpID"]);
textBox3.Text = Convert.ToString(drow["EmpName"]);
textBox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button4_Click(object sender, EventArgs e)
{
    //last
    int n = dt.Rows.Count - 1;
    textBox1.Text = Convert.ToString(n);
    drow = dt.Rows[n];
    textBox2.Text = Convert.ToString(drow["EmpID"]);
    textBox3.Text = Convert.ToString(drow["EmpName"]);
    textBox4.Text = Convert.ToString(drow["Salary"]);
}

private void Button5_Click(object sender, EventArgs e)
{
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    button5.Visible = false;
    button6.Visible = true;
    textBox2.ReadOnly = false;
    textBox2.Focus();
}
```

```

private void Button6_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "insert into Employees values(@empid,
@empname, @salary");
    cmd.Connection = cn;
    p1.ParameterName = "@empid";
    p2.ParameterName = "@empname";
    p3.ParameterName = "@salary";
    p1.Value = textbox2.Text;
    p2.Value = textbox3.Text;
    p3.Value = textbox4.Text;

    //calling methods
    cmd.Parameters.Add(p1);
    cmd.Parameters.Add(p2);
    cmd.Parameters.Add(p3);
    cn.Open();
    int m = cmd.ExecuteNonQuery();
    cn.Close();

    textbox2.ReadOnly = true;
    string msg = m + " rows inserted";

```

```
MessageBox.Show(msg);

loaddata();

//show inserted record
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    if (Convert.ToString(drow["empid"]) == textbox2.Text)
    {
        textbox1.Text = Convert.ToString(i);
        textbox2.Text = Convert.ToString(drow["EmpID"]);
        textbox3.Text = Convert.ToString(drow["EmpName"]);
        textbox4.Text = Convert.ToString(drow["Salary"]);
    }
}

button5.Visible = true;
button6.Visible = false;
}

private void Button7_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1, p2, p3;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();

    //calling properties
```

```

cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "update Employees set empname=@empname,
salary=@salary where empid=@empid";
cmd.Connection = cn;
p1.ParameterName = "@empid";
p2.ParameterName = "@empname";
p3.ParameterName = "@salary";
p1.Value = textbox2.Text;
p2.Value = textbox3.Text;
p3.Value = textbox4.Text;

//calling methods
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);
cn.Open();
int m = cmd.ExecuteNonQuery();
cn.Close();

string msg = m + " rows updated";
MessageBox.Show(msg);

loaddata();

//show updated record
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    if (Convert.ToString(drow["empid"]) == textbox2.Text)
    {
        textbox1.Text = Convert.ToString(i);
        textbox2.Text = Convert.ToString(drow["EmpID"]);
        textbox3.Text = Convert.ToString(drow["EmpName"]);
        textbox4.Text = Convert.ToString(drow["Salary"]);
    }
}

```

```
button5.Visible = true;
button6.Visible = false;
}

private void Button8_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlParameter p1;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();

    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "delete from Employees where empid=@empid";
    cmd.Connection = cn;
    p1.ParameterName = "@empid";
    p1.Value = textbox2.Text;

    //calling methods
    cmd.Parameters.Add(p1);
    cn.Open();
    int m = cmd.ExecuteNonQuery();
    cn.Close();

    string msg = m + " rows deleted";
    MessageBox.Show(msg);

    loaddata();

    //previous
}
```

```
int n = Convert.ToInt32(textBox1.Text);
n--;
if (n == -1)
{
    n = 0;
    MessageBox.Show("Already at first record");
}
textBox1.Text = Convert.ToString(n);
drow = dt.Rows[n];
textBox2.Text = Convert.ToString(drow["EmpID"]);
textBox3.Text = Convert.ToString(drow["EmpName"]);
textBox4.Text = Convert.ToString(drow["Salary"]);
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Record No:

Emp ID:

Emp Name:

Salary:

Buttons:

- First
- Previous
- Next
- Last
- Create
- Update
- Delete

Try to use all the buttons.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – Registration Form - Example

Creating Database

- Note: Ignore this step, if you have created “useraccountsdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database useraccountsdatabase

go

use useraccountsdatabase

go

create table Users(
userid int primary key identity(1,1),
username nvarchar(max),
password nvarchar(max),

```
email nvarchar(max))
```

```
go
```

```
insert into Users values('scott', 'scott123', 'scott@gmail.com')
```

```
insert into Users values('john', 'john123', 'john@gmail.com')
```

```
insert into Users values('jones', 'jones123', 'jones@gmail.com')
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “RegistrationFormExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “RegistrationFormExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
```

```
namespace RegistrationFormExample
```

```
{
```

```
    public partial class Form1 : Form
    {
```

```
Label label1, label2, label3, label4;
TextBox textbox1, textbox2, textbox3, textbox4;
Button button1;

public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(650, 400);
    this.Text = "Registration Form";
    this.StartPosition = FormStartPosition.CenterScreen;

    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = "Username: ";
    label1.Location = new Point(50, 50);
    this.Controls.Add(label1);

    /* label2 */
    label2 = new Label();
    label2.AutoSize = true;
    label2.Text = "Password: ";
    label2.Location = new Point(50, 100);
    this.Controls.Add(label2);

    /* label3 */
    label3 = new Label();
    label3.AutoSize = true;
    label3.Text = "Confirm Password: ";
    label3.Location = new Point(50, 150);
    this.Controls.Add(label3);

    /* label4 */
    label4 = new Label();
```

```
label4.AutoSize = true;
label4.Text = "Email:";
label4.Location = new Point(50, 200);
this.Controls.Add(label4);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(300, 50);
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(300, 100);
textbox2.PasswordChar = '*';
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(300, 150);
textbox3.PasswordChar = '*';
this.Controls.Add(textbox3);

/* textbox4 */
textbox4 = new TextBox();
textbox4.Size = new Size(300, 50);
textbox4.Location = new Point(300, 200);
this.Controls.Add(textbox4);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Register";
button1.Location = new Point(300, 250);
button1.Click += Button1_Click;
```

```

    this.AcceptButton = button1;
    this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
{
    if (textbox1.Text == "" || textbox2.Text == "" || textbox3.Text == "" ||
    textbox4.Text == "")
    {
        MessageBox.Show("Field can't be blank.");
    }
    else if (textbox2.Text != textbox3.Text)
    {
        MessageBox.Show("Password and confirm password do not
match.");
    }
    else
    {
        try
        {
            //create reference variables
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1, p2, p3;

            //create objects
            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
            p2 = new SqlParameter();
            p3 = new SqlParameter();

            //calling properties
            cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=useraccountsdatabase";
            cmd.CommandText = "insert into Users values(@username,
@password, @email)";
        }
    }
}

```

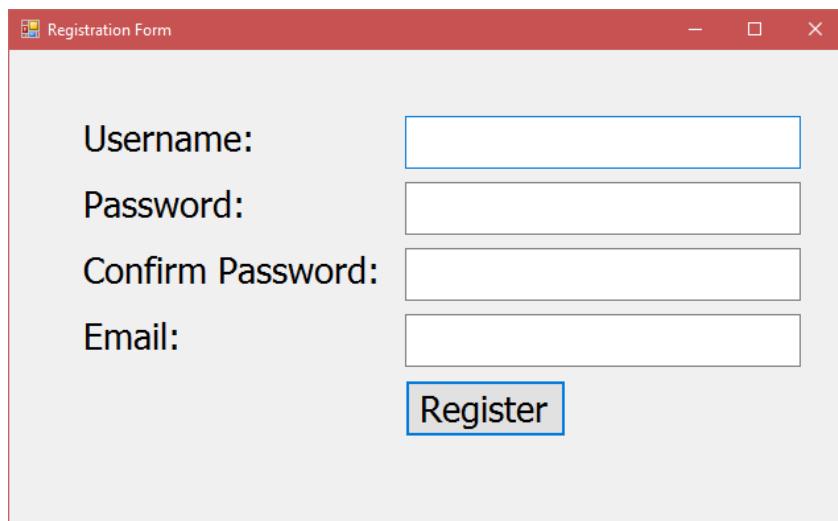
```
cmd.Connection = cn;
p1.ParameterName = "@username";
p2.ParameterName = "@password";
p3.ParameterName = "@email";
p1.Value = textbox1.Text;
p2.Value = textbox2.Text;
p3.Value = textbox4.Text;

//calling methods
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
cmd.Parameters.Add(p3);
cn.Open();
int n = cmd.ExecuteNonQuery();
cn.Close();
string msg;
if (n == 1)
    msg = "Successfully Registered";
else
    msg = "Registration failed";
MessageBox.Show(msg);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some user details and click on “Register”. The data will be saved in the database.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – Login Form - Example

Creating Database

- Note: Ignore this step, if you have created “useraccountsdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database useraccountsdatabase
```

```
go
```

```
use useraccountsdatabase
```

```
go
```

```
create table Users(  

userid int primary key identity(1,1),  

username nvarchar(max),  

password nvarchar(max),  

email nvarchar(max))
```

```
go
```

```
insert into Users values('scott', 'scott123', 'scott@gmail.com')
```

```
insert into Users values('john', 'john123', 'john@gmail.com')
```

```
insert into Users values('jones', 'jones123', 'jones@gmail.com')
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.

- Type the project name as “LoginExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LoginExample”.
- Click on OK.
- Open Solution Explorer. Right click on the project (LoginExample) and click on “Add” – “New Item” – “Windows Forms” – “Windows Form”. Type the file name as “Form2.cs”. Click on “Add”.
- Open Solution Explorer. Right click on the project (LoginExample) and click on “Add” – “New Item” – “Code” – “Class”. Type the file name as “Common.cs”. Click on “Add”.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace LoginExample
{
    public partial class Form1 : Form
    {
        Label label1, label2;
        TextBox textbox1, textbox2;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
        }
}
```

```
this.Size = new Size(680, 300);
this.Text = "Login Form";
this.StartPosition = FormStartPosition.CenterScreen;

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = "Username: ";
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Password: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* textBox1 */
textBox1 = new TextBox();
textBox1.Size = new Size(300, 50);
textBox1.Location = new Point(300, 50);
this.Controls.Add(textBox1);

/* textBox2 */
textBox2 = new TextBox();
textBox2.Size = new Size(300, 50);
textBox2.Location = new Point(300, 100);
textBox2.PasswordChar = '*';
this.Controls.Add(textBox2);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Login";
button1.Location = new Point(300, 150);
button1.Click += Button1_Click;
```

```
this.AcceptButton = button1;
this.Controls.Add(button1);
}

private void Button1_Click(object sender, EventArgs e)
{
    try
    {
        //get values from textboxes
        string uname = textbox1.Text;
        string pwd = textbox2.Text;

        //create reference variables
        SqlConnection cn;
        SqlCommand cmd;
        SqlParameter p1, p2;
        SqlDataAdapter adp;
        DataSet ds;
        DataTable dt;
        DataRow drow;

        //create objects
        cn = new SqlConnection();
        cmd = new SqlCommand();
        p1 = new SqlParameter();
        p2 = new SqlParameter();
        adp = new SqlDataAdapter();
        ds = new DataSet();

        //call properties
        cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=useraccountsdatabase";
        cmd.CommandText = "select * from users where username=@username and password=@password";
        cmd.Connection = cn;
        p1.ParameterName = "@username";
        p1.Value = uname;
```

```
p2.ParameterName = "@password";
p2.Value = pwd;
adp.SelectCommand = cmd;

//call methods
cmd.Parameters.Add(p1);
cmd.Parameters.Add(p2);
adp.Fill(ds);

//read data
dt = ds.Tables[0];

if (dt.Rows.Count == 1)
{
    object obj1;
    drow = dt.Rows[0];
    obj1 = drow["email"];
    string email = Convert.ToString(obj1);

    Common.CurrentEmail = email;

    Form2 f = new Form2();
    f.Show();
    this.Hide();
}
else
{
    MessageBox.Show("Invalid username or password, please try
again!");
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
}
```

}

Form2.cs

```
using System;
using System.Windows.Forms;

namespace LoginExample
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            string msg = "Welcome to " + Common.CurrentEmail;
            //label1.Text = msg;
        }

        private void Form2_FormClosing(object sender,
FormClosingEventArgs e)
        {
            Environment.Exit(0);
        }
    }
}
```

Common.cs

```
using System;

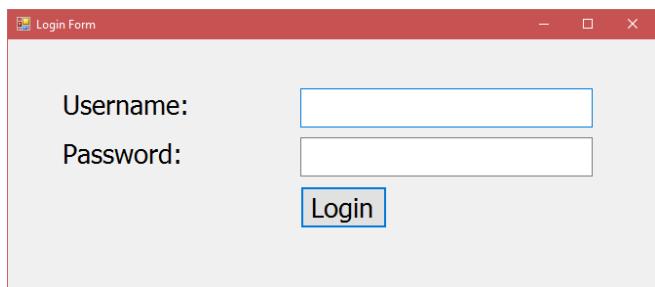
namespace LoginExample
{
    static class Common
    {
        public static string CurrentEmail { get; set; }
    }
}
```

```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Enter “scott” and “scott123” as username and password and click on “Login”.

ADO.NET – Transactions - Example**Creating Database**

- Note: Ignore this step, if you have created “transactionsdatabase” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database transactionsdatabase

go

use transactionsdatabase

go

CREATE TABLE AccountsTable(

AccountNo int primary key,

AccountHolderName nvarchar(40),

Balance decimal)

GO

INSERT INTO AccountsTable VALUES (101, 'scott', 10000)

INSERT INTO AccountsTable VALUES (102, 'allen', 5000)

GO

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “TransactionsExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “TransactionsExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;

namespace TransactionsExample
{
    public partial class Form1 : Form
    {
        Button button1;
```

```
public Form1()
{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(500, 300);
    this.Text = "ADO.NET Transactions";
    this.StartPosition = FormStartPosition.CenterScreen;

    /* button1 */
    button1 = new Button();
    button1.Text = "Start Transaction";
    button1.AutoSize = true;
    button1.Location = new Point(130, 100);
    button1.Click += button1_Click;
    this.Controls.Add(button1);
}

private void button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlTransaction transaction;
    SqlCommand cmd1, cmd2;

    //create objects
    cn = new SqlConnection();
    cmd1 = new SqlCommand();
    cmd2 = new SqlCommand();

    //call properties
    cn.ConnectionString = "data source=localhost; integrated security=yes; initial catalog=transactionsdatabase";
    cmd1.CommandText = "update AccountsTable set Balance=Balance-1000 where AccountNo=101";
}
```

```

cmd2.CommandText = "update AccountsTable set
Balance=Balance+1000 where AccountNo=102";
cmd1.Connection = cn;
cmd2.Connection = cn;

//call methods
cn.Open();
transaction = cn.BeginTransaction();
cmd1.Transaction = transaction;
cmd2.Transaction = transaction;

try
{
    cmd1.ExecuteNonQuery();
    MessageBox.Show("First operation done.");

    cmd2.ExecuteNonQuery();
    MessageBox.Show("Second operation done.");
    transaction.Commit(); //save data

    MessageBox.Show("Transaction Complete");
}

catch (Exception)
{
    transaction.Rollback(); //first operation will be rollback
    MessageBox.Show("Rollback done!");
}

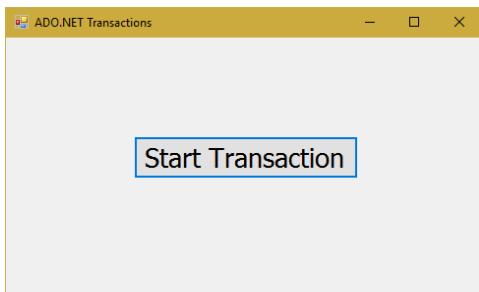
cn.Close();
}
}
}

```

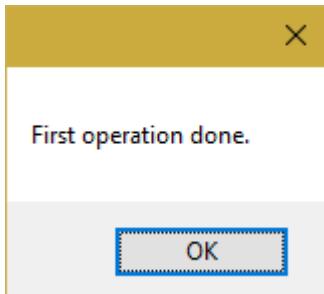
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

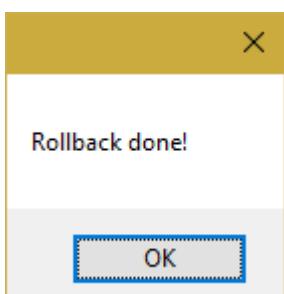
Output



Click on “Start Transaction”.



Click on OK.



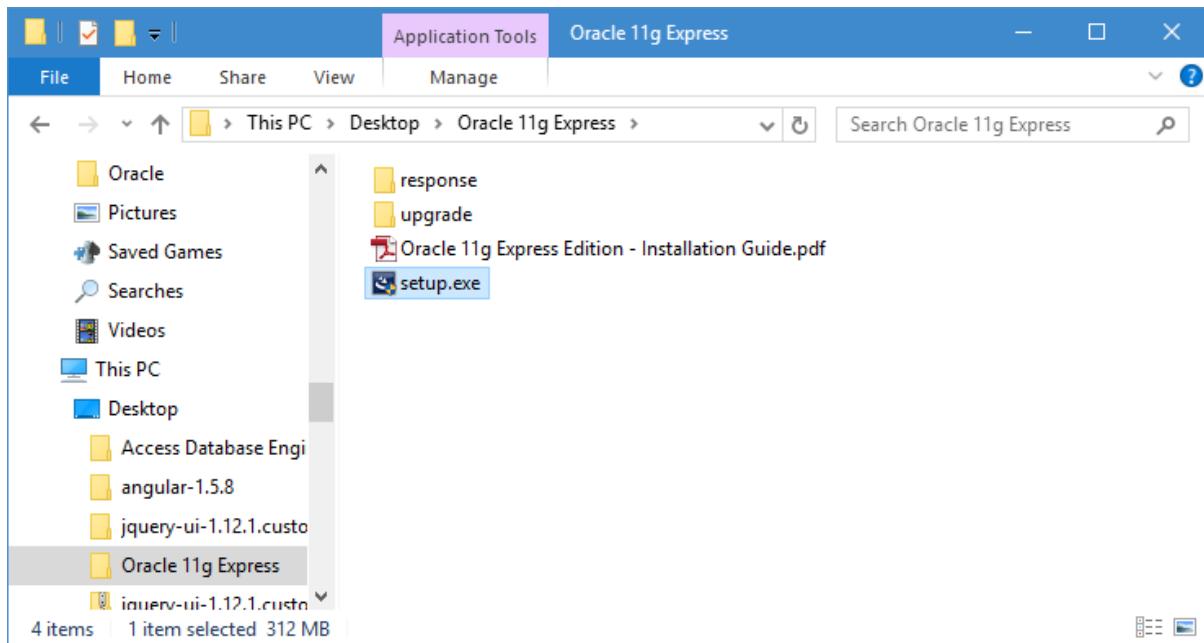
Note: If any database connection problem, it shows exception (run time error).

ADO.NET – Oracle - Example

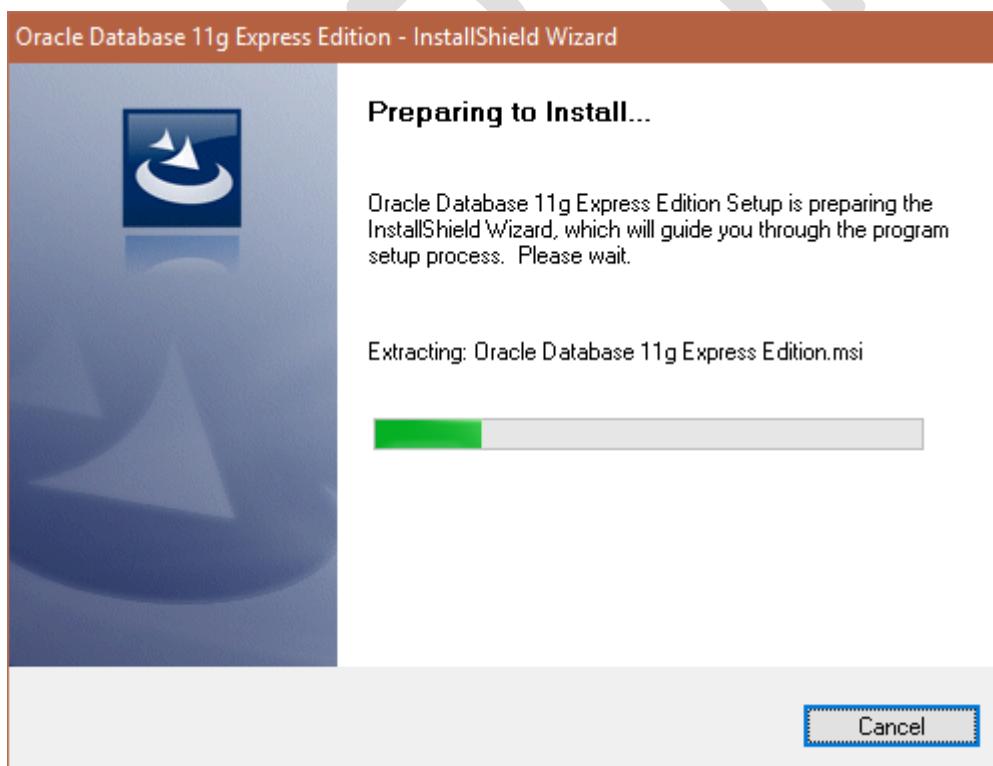
Installing Oracle Database 11g Expression Edition

- Note: Ignore this step, if you have installed “Oracle 11g Express” already.
- You can download Oracle 11g Express Edition at:
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

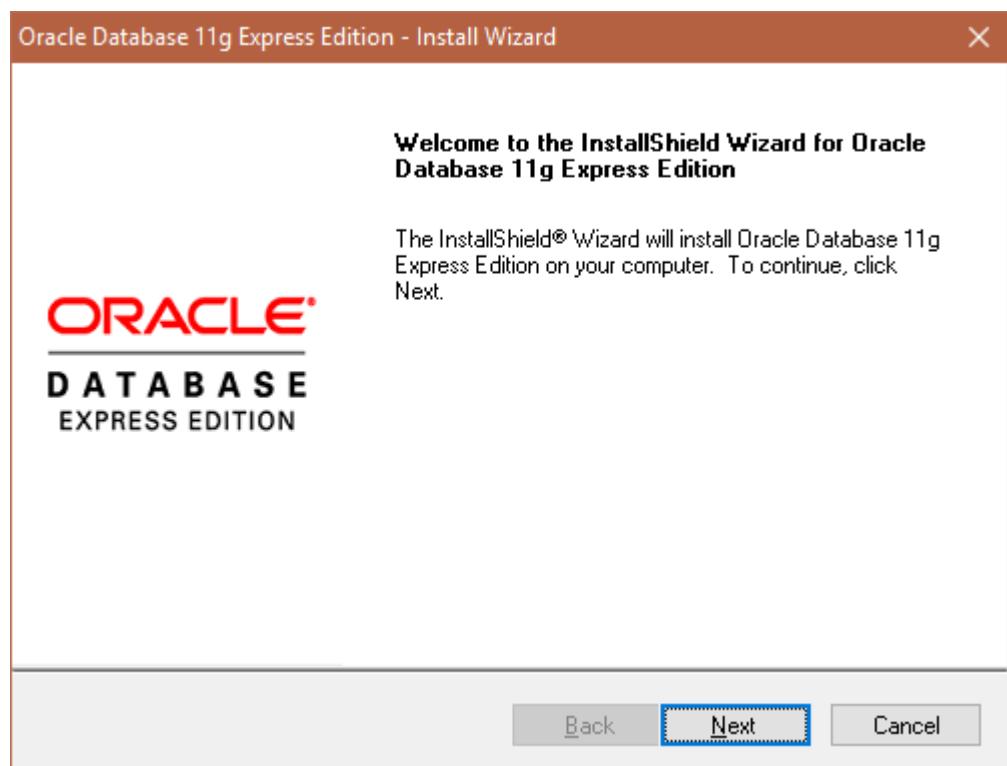
- Open “Oracle 11g Express” folder.



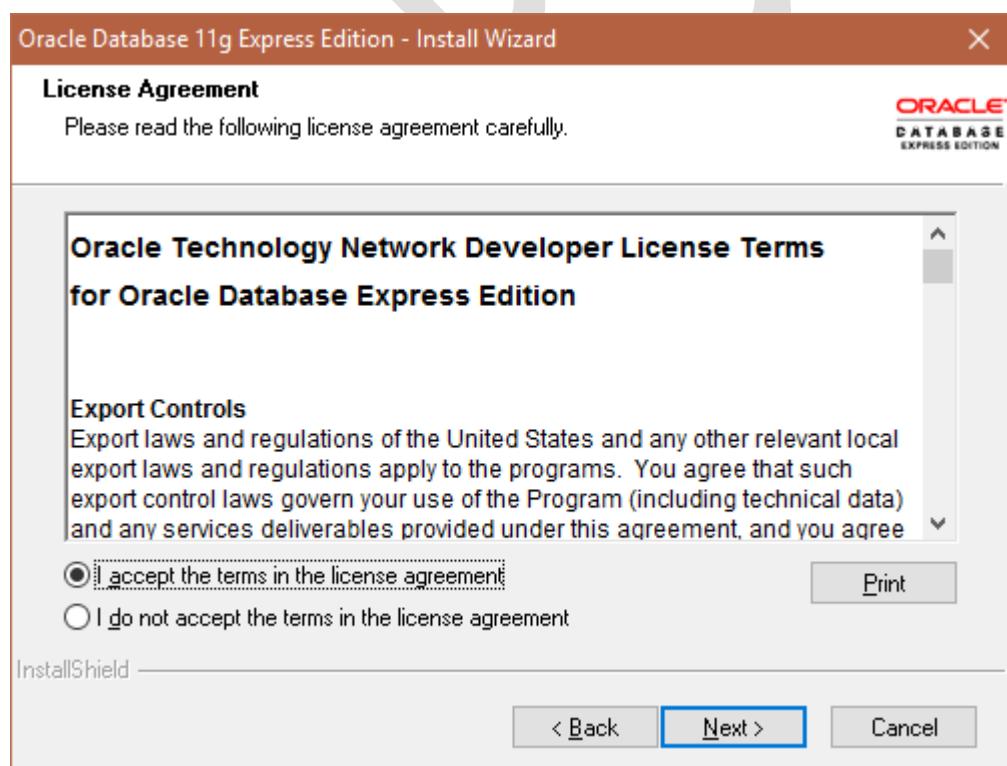
- Double click on “setup.exe”.
- Click on “Yes”.



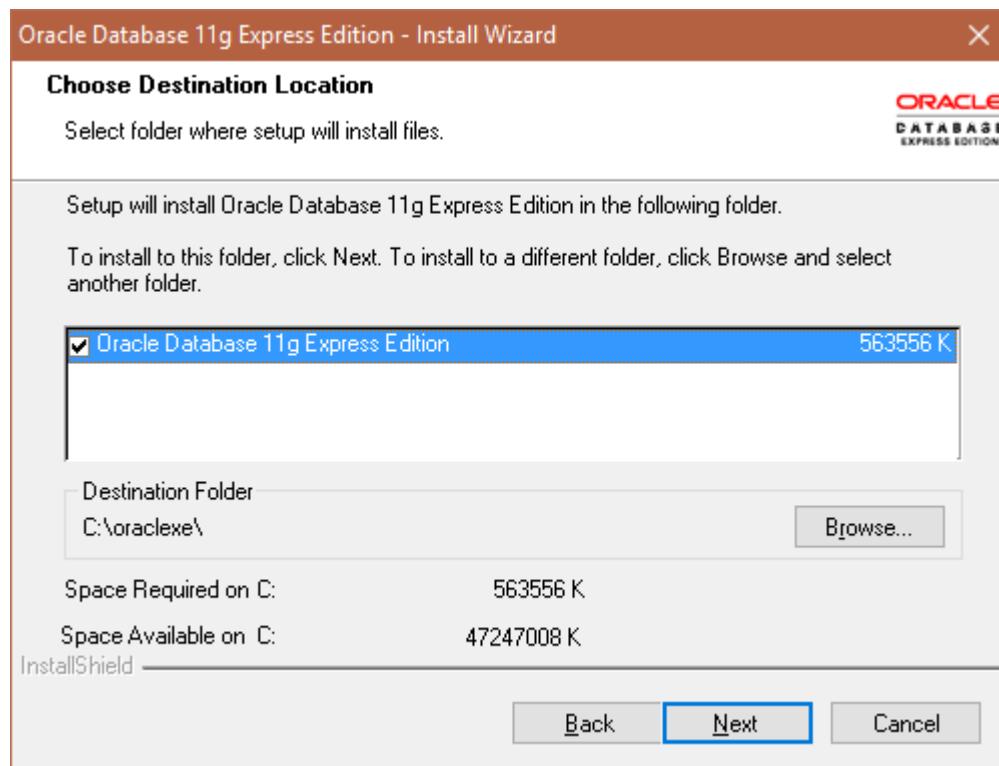
- Please wait while it loads...



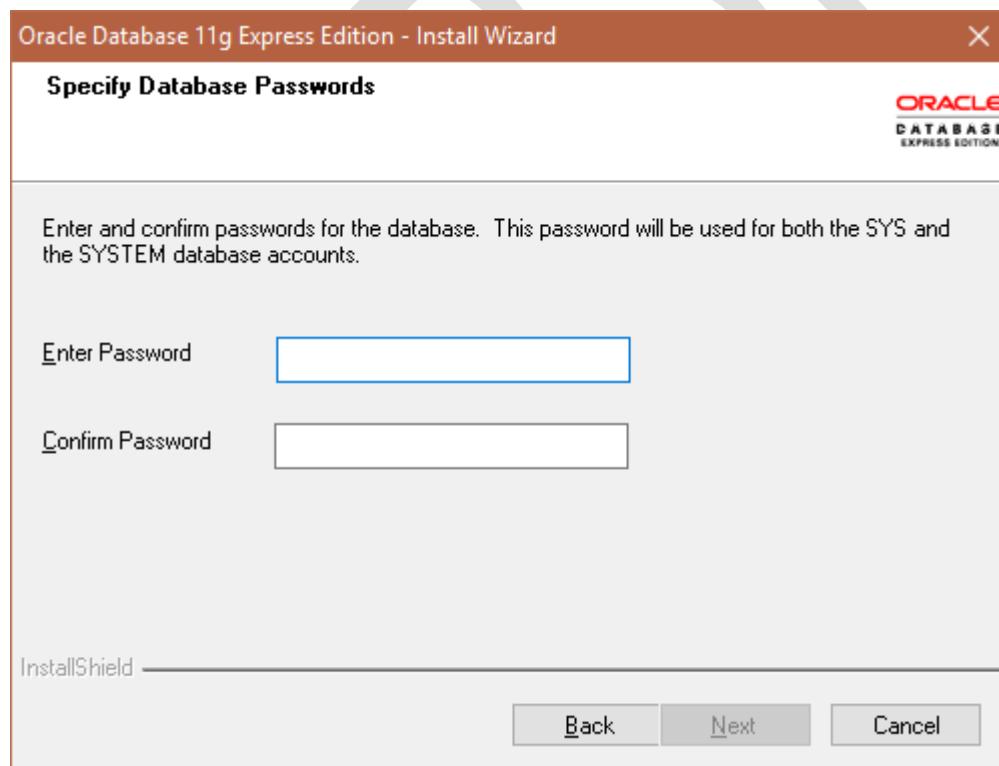
- Click on "Next".
- Click on "I accept the terms in the license agreement".



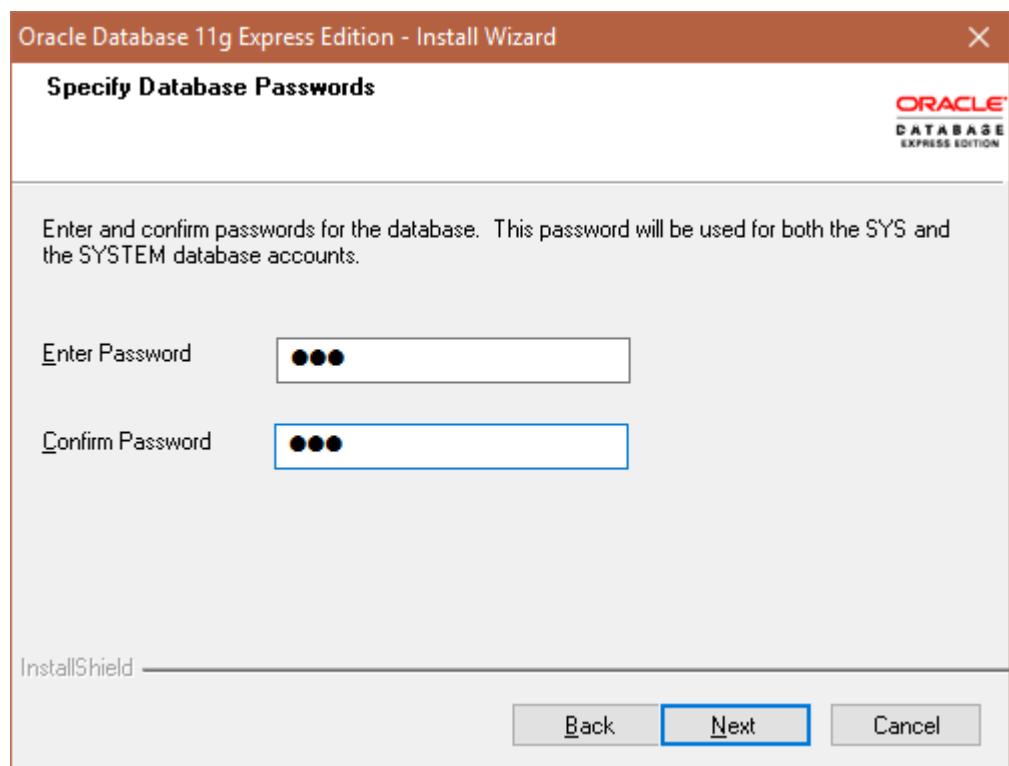
- Click on "Next".



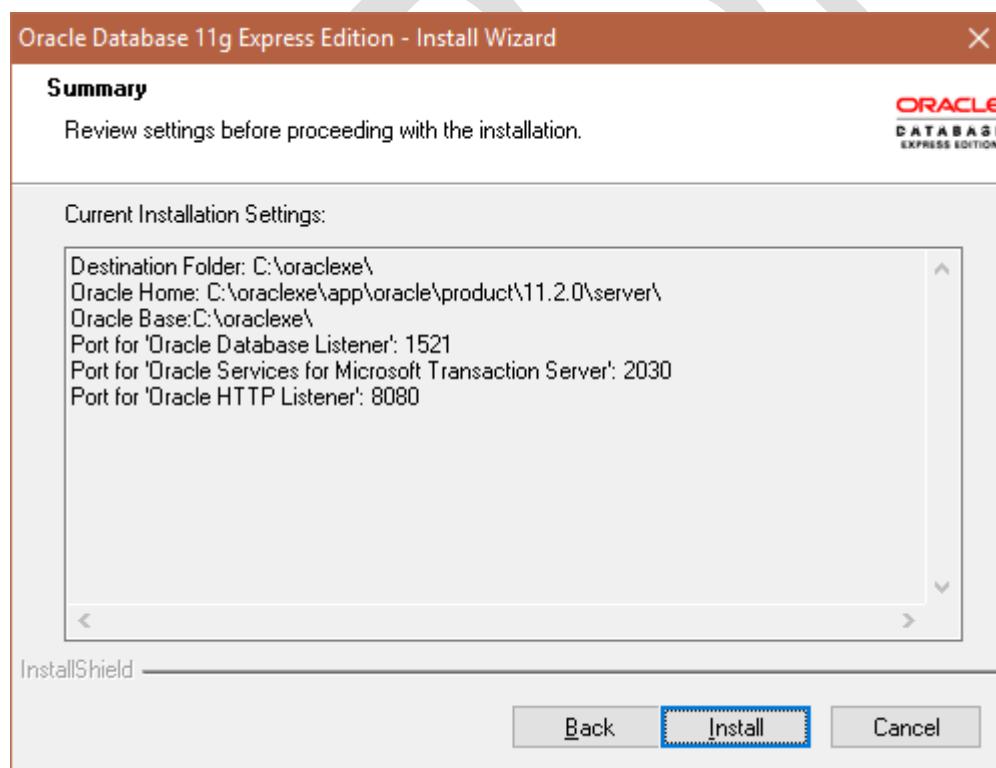
- Click on “Next”.



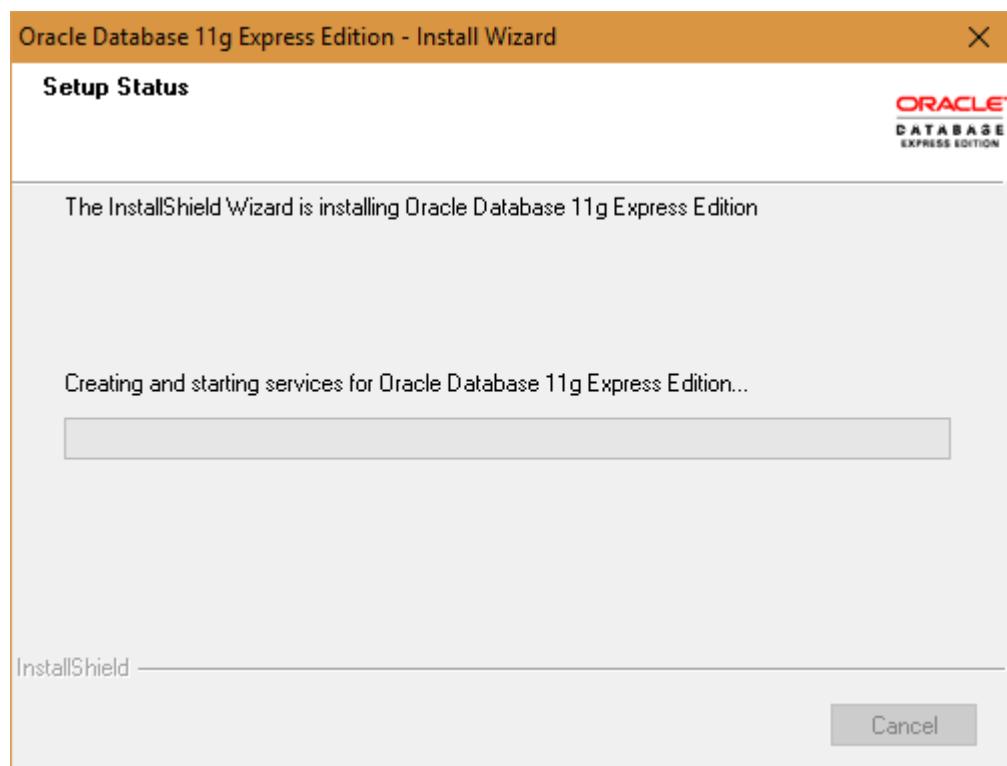
- Enter the password as “123”.
- Enter the confirm password “123”.



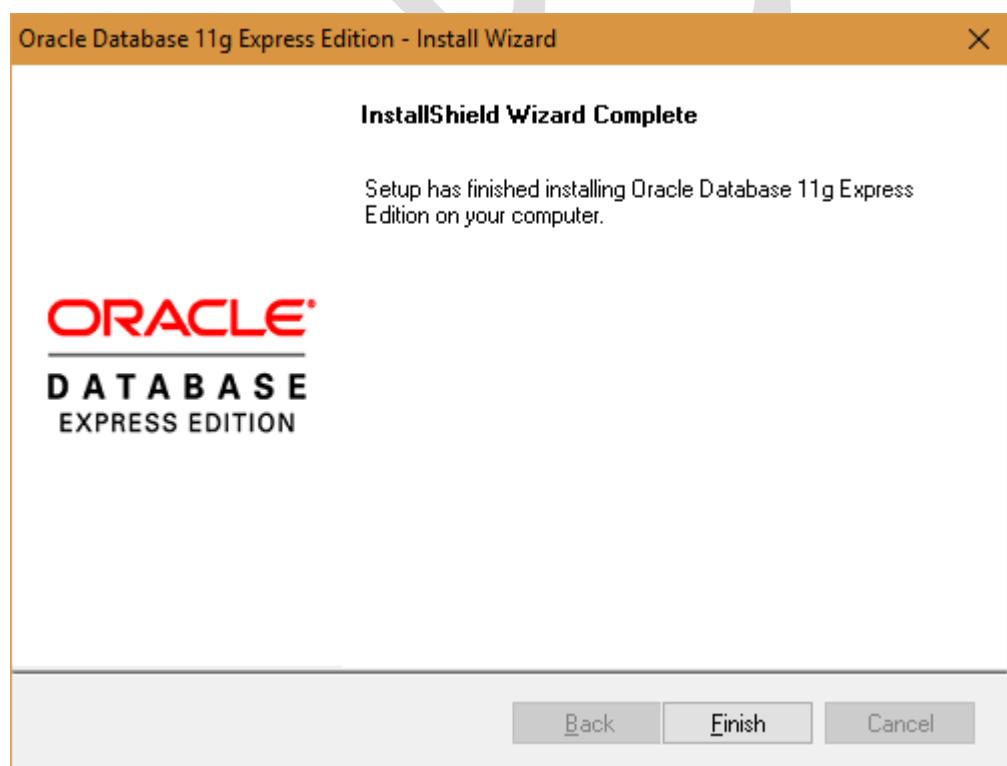
- Click on "Next".



- Click on "Install".



- Installation may take around 10 minutes.
 - Click on OK if it shows one or two errors while installing.

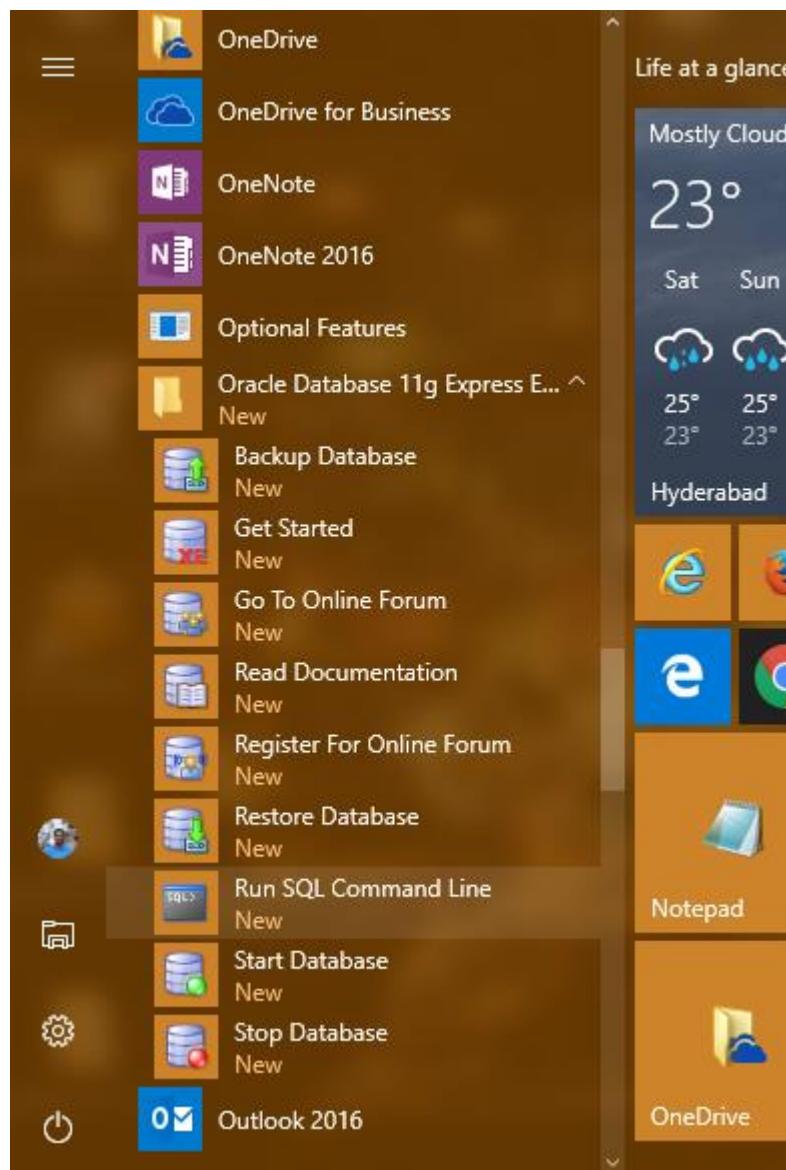


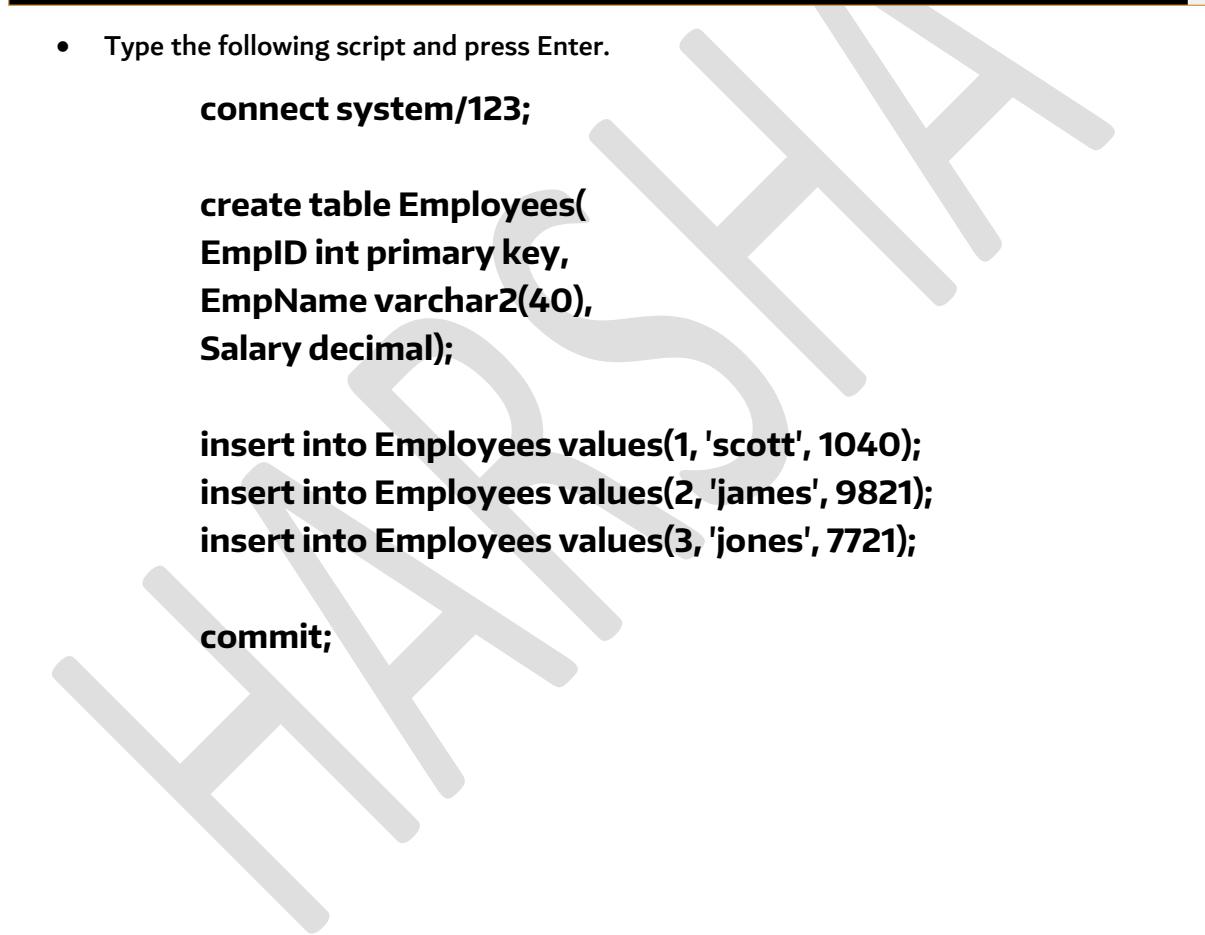
- Click on "Finish".

- Restart the PC.

Creating table in Oracle

- Note: Ignore this step, if you have created “employees” table in Oracle already.
- Go to “Start” – “Oracle 11g Express Edition” – “Run SQL Command Line”.





```
Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 24 12:47:33 2016
Copyright (c) 1982, 2010, Oracle. All rights reserved.
SQL> -
```

- Type the following script and press Enter.

```
connect system/123;
```

```
create table Employees(
EmpID int primary key,
EmpName varchar2(40),
Salary decimal);
```

```
insert into Employees values(1, 'scott', 1040);
insert into Employees values(2, 'james', 9821);
insert into Employees values(3, 'jones', 7721);
```

```
commit;
```



```
Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 24 12:47:33 2016
Copyright (c) 1982, 2010, Oracle. All rights reserved.

SQL> connect system/123;
Connected.
SQL>
SQL> create table Employees(
  2  EmpID int primary key,
  3  EmpName varchar2(40),
  4  Salary decimal);

Table created.

SQL>
SQL> insert into Employees values(1, 'scott', 1040);

1 row created.

SQL> insert into Employees values(2, 'james', 9821);

1 row created.

SQL> insert into Employees values(3, 'jones', 7721);

1 row created.

SQL>
SQL> commit;

Commit complete.

SQL>
```

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “OracleExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OracleExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.OleDb;
using System.Data;

namespace OracleExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "Oracle";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.AutoScroll = true;
            this.Load += Form1_Load;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
/* create reference variables */
OleDbConnection cn;
OleDbCommand cmd;
OleDbDataAdapter adp;
DataSet ds;
DataTable dt;
DataRow drow;

/* create objects */
cn = new OleDbConnection();
cmd = new OleDbCommand();
adp = new OleDbDataAdapter();
ds = new DataSet();

/* call properties */
cn.ConnectionString = "user id=system; password=123;
provider=msdaora.1";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
int n = 50;
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;
```

```
eid = Convert.ToInt32(obj1);
ename = Convert.ToString(obj2);
sal = Convert.ToDecimal(obj3);

Label label1, label2, label3;

/* label1 */
label1 = new Label();
label1.AutoSize = true;
label1.Text = Convert.ToString(eid);
label1.Location = new Point(50, n);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = ename;
label2.Location = new Point(150, n);
this.Controls.Add(label2);

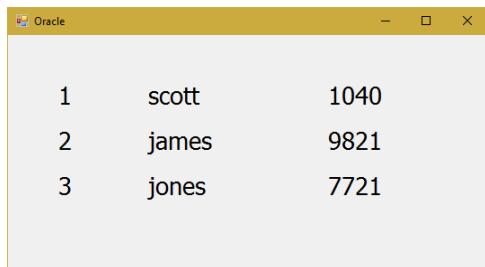
/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = Convert.ToString(sal);
label3.Location = new Point(350, n);
this.Controls.Add(label3);

/* go to next row */
n += 50;
}

}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

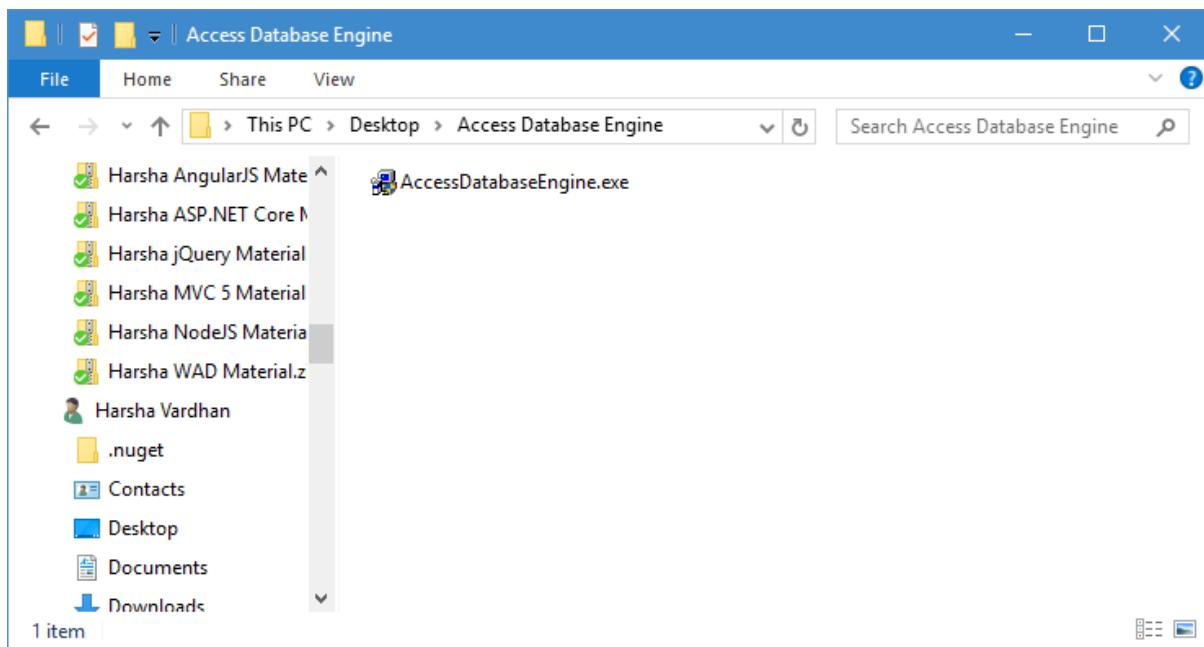
1	scott	1040
2	james	9821
3	jones	7721

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – MS Access - Example

Installing Access Database Engine

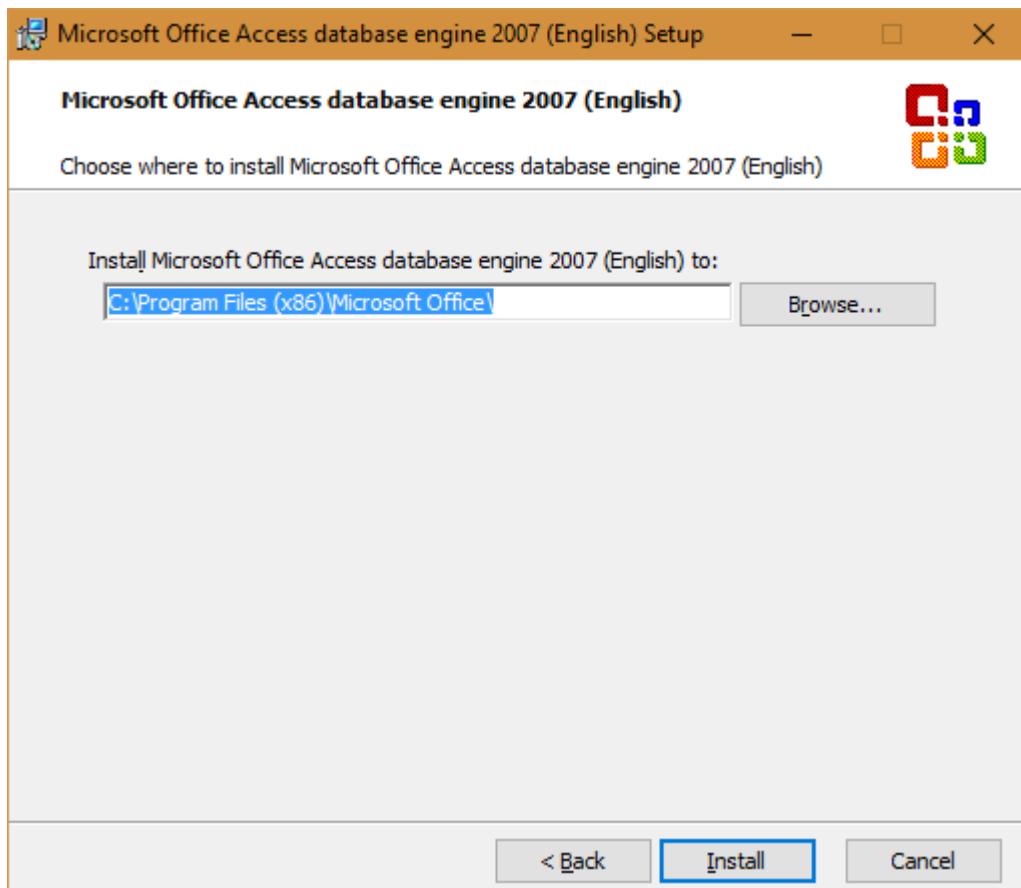
- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at:
<https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”.
- **Note:** Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>
- To install “Access Database Engine”, go to “Access Database Engine” folder.



- Double click on “AccessDatabaseEngine”.
- Click on “Yes”.



- Check the checkbox “I accept the terms in the License Agreement”.
- Click on “Next”.



- Click on “Install”.
- Installation may take around 1 or 2 minutes.

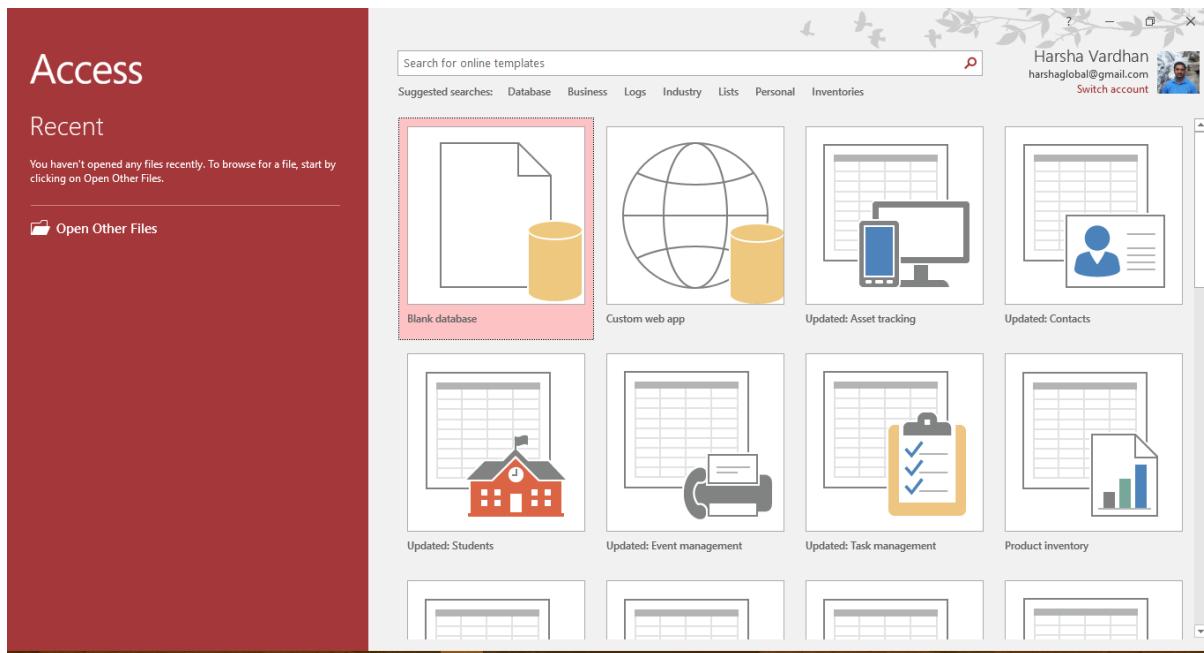


- Click on OK.

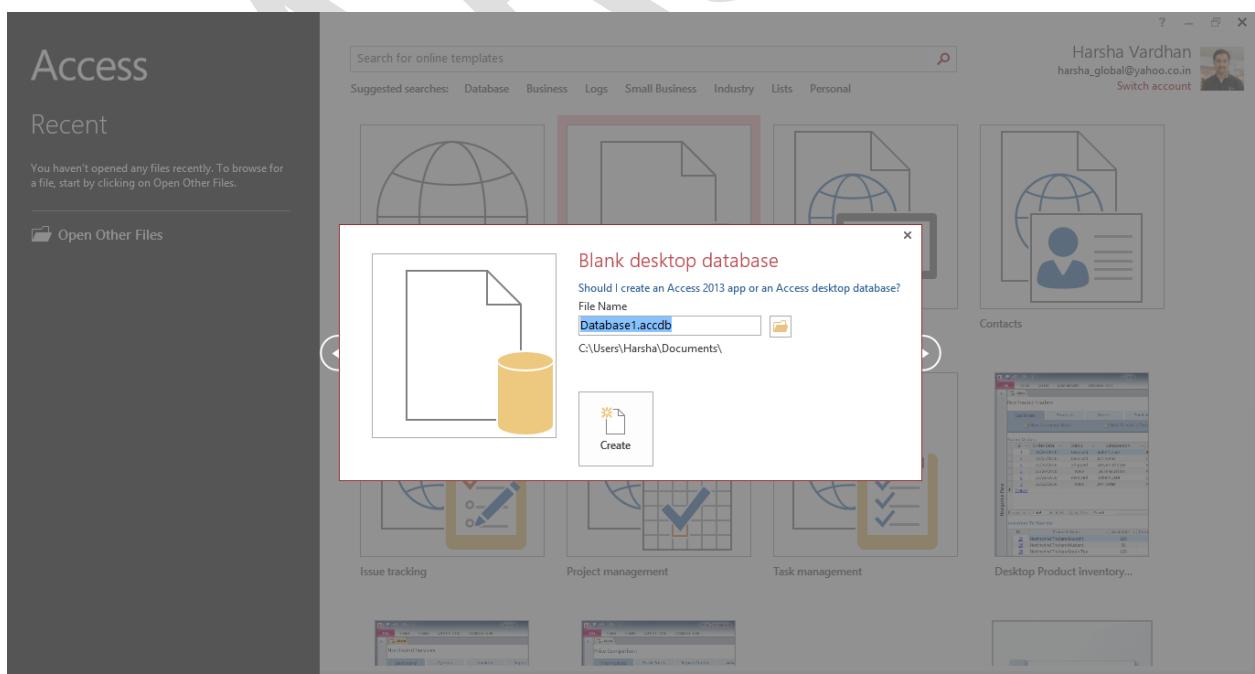
HARSHA

Creating table in MS Access

- Go to “Start” – “Access 2016”.

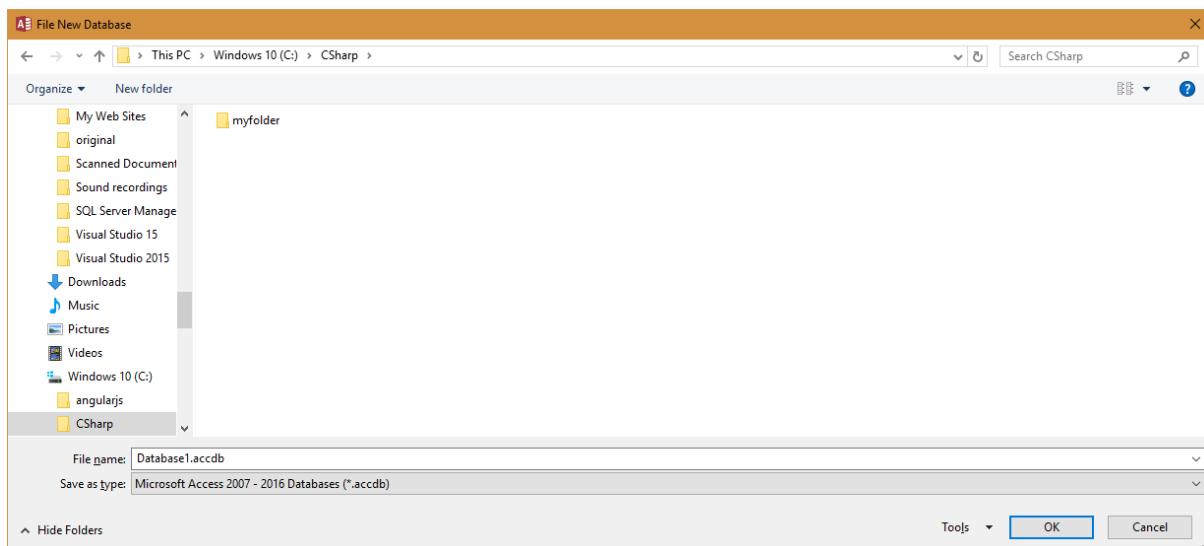


- Click on “Blank desktop database”.
- Type the file name as “Database1.accdb”.

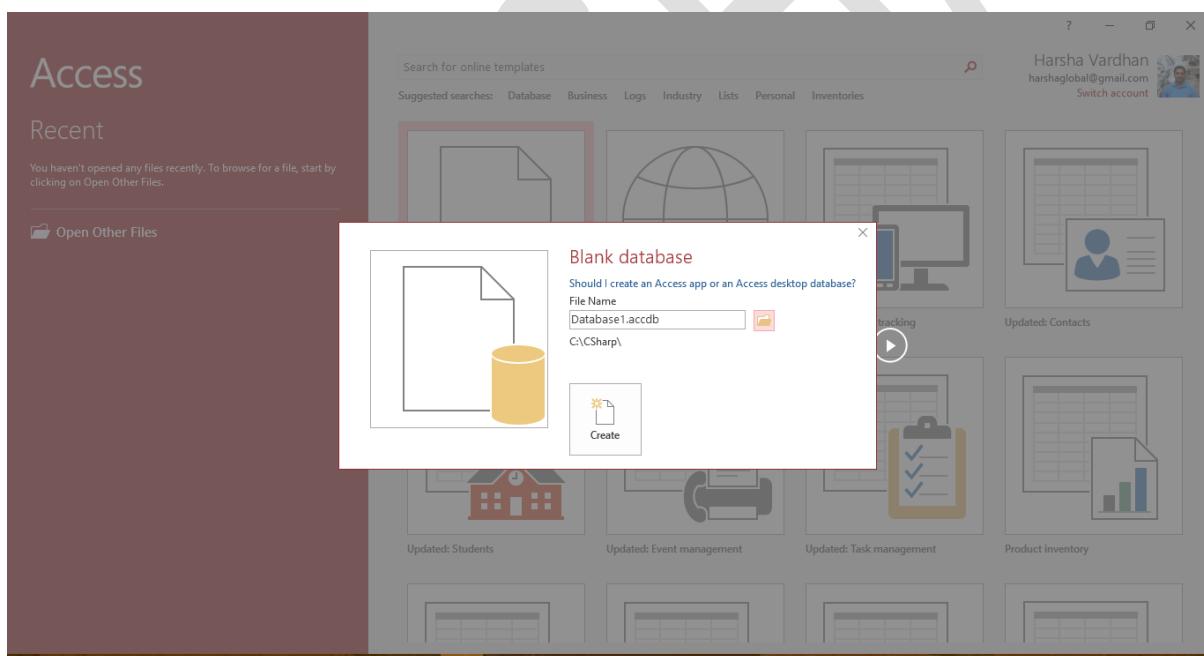


- Click on the folder icon and select “C:\CSharp” folder.

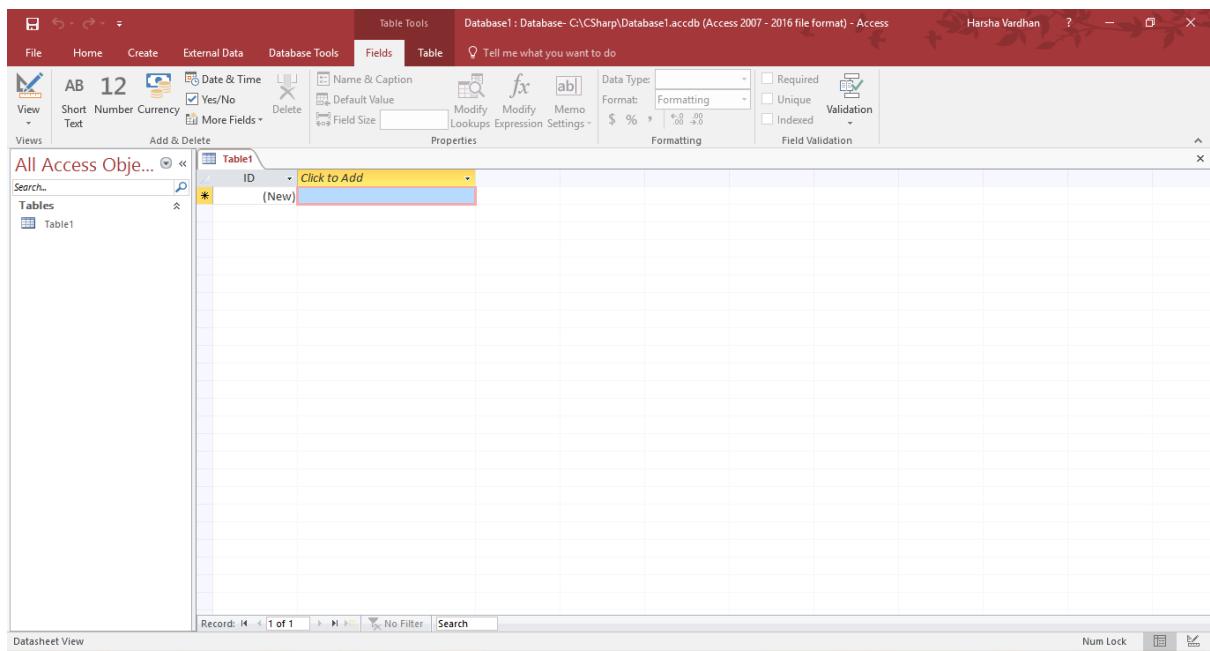
C#.NET 8.0



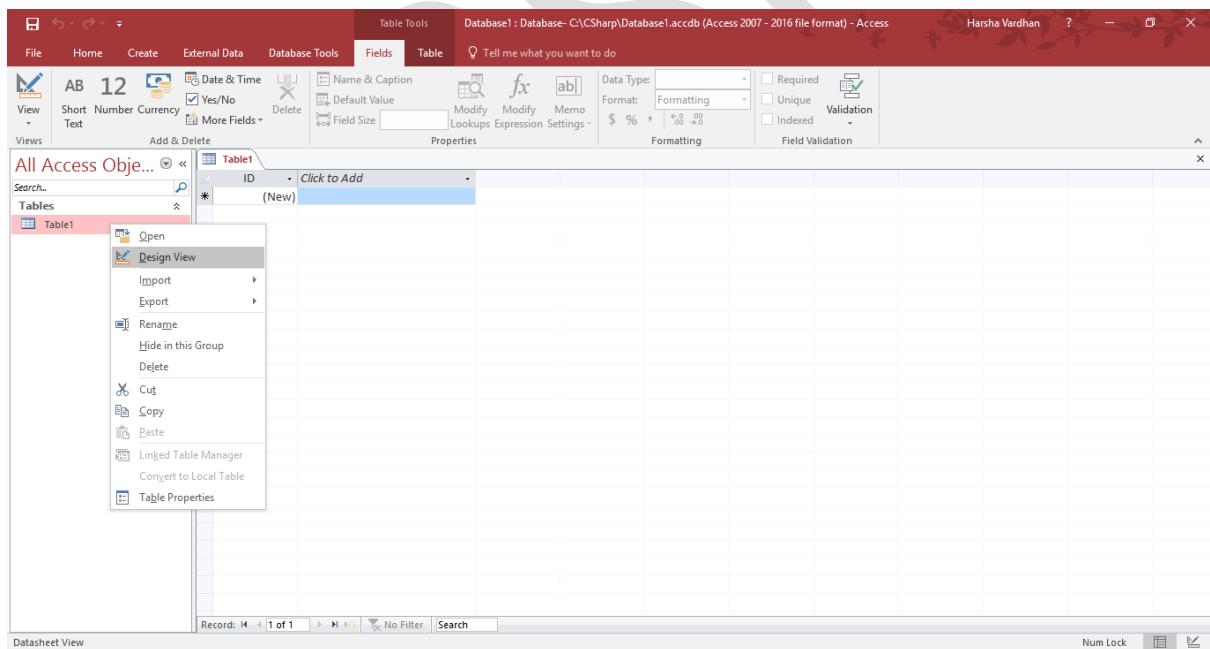
- Click on OK.



- Click on "Create".



- Right click on “Table1” and click on “Design View”.



- Type the table name as “Employees”.



- Click on OK.
- Type the table structure as follows:

A screenshot of Microsoft Access in Design view. The ribbon shows 'Database Tools' and 'Design'. The left pane shows 'All Access Objects' with 'Tables' expanded and 'Employees' selected. The main area displays the 'Employees' table structure:

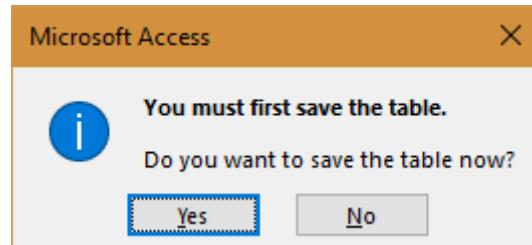
Field Name	Data Type	Description (Optional)
EmpID	AutoNumber	
EmpName	Short Text	
Salary	Number	

The status bar at the bottom indicates 'Design view. F6 = Switch panes. F1 = Help.'

- Right click "Employees" and click on "Open".

A screenshot of Microsoft Access showing a context menu for the 'Employees' table. The menu items include 'Open' (highlighted with a blue border), 'Design View', 'Import', 'Export', 'Rename', 'Hide in this Group', 'Delete', 'Cut', 'Copy', 'Paste', 'Linked Table Manager', 'Convert to Local Table', and 'Table Properties'. The main area shows the same table structure as the previous screenshot. The status bar at the bottom indicates 'Design view. F6 = Switch panes. F1 = Help.'

- Click on “Yes”.

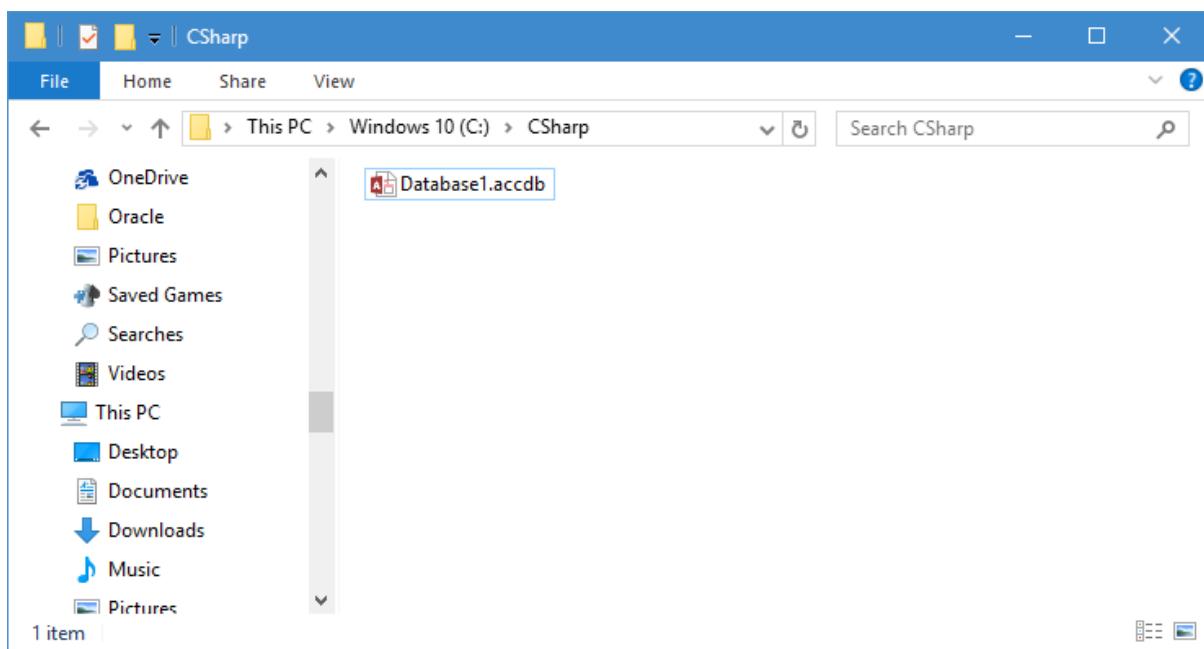


- Type the data as follows:

A screenshot of the Microsoft Access application. The ribbon is visible with "Home" selected. A table named "Employees" is open, showing the following data:

EmpID	EmpName	Salary	Action
1	Scott	5000	
2	Allen	6000	
3	Jones	7000	
*	(New)	0	

- Save the file.
- Close “Microsoft Access 2016”.
- Make sure “Database1.accdb” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MSAccessExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MSAccessExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;  
using System.Windows.Forms;
```

```
using System.Drawing;
using System.Data.OleDb;
using System.Data;

namespace MSAccessExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "MS Access";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.AutoScroll = true;
            this.Load += Form1_Load;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            /* create reference variables */
            OleDbConnection cn;
            OleDbCommand cmd;
            OleDbDataAdapter adp;
            DataSet ds;
            DataTable dt;
            DataRow drow;

            /* create objects */
            cn = new OleDbConnection();
            cmd = new OleDbCommand();
            adp = new OleDbDataAdapter();
            ds = new DataSet();
        }
    }
}
```

```

/* call properties */
cn.ConnectionString = @"provider=Microsoft.Ace.Oledb.12.0; data
source=C:\CSharp\Database1.accdb";
cmd.CommandText = "select * from Employees";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
int n = 50;
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];

    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid;
    string ename;
    decimal sal;

    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);

    Label label1, label2, label3;

    /* label1 */
    label1 = new Label();
    label1.AutoSize = true;
    label1.Text = Convert.ToString(eid);
    label1.Location = new Point(50, n);
    this.Controls.Add(label1);
}

```

```
/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = ename;
label2.Location = new Point(150, n);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = Convert.ToString(sal);
label3.Location = new Point(350, n);
this.Controls.Add(label3);

/* go to next row */
n += 50;
}

}
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

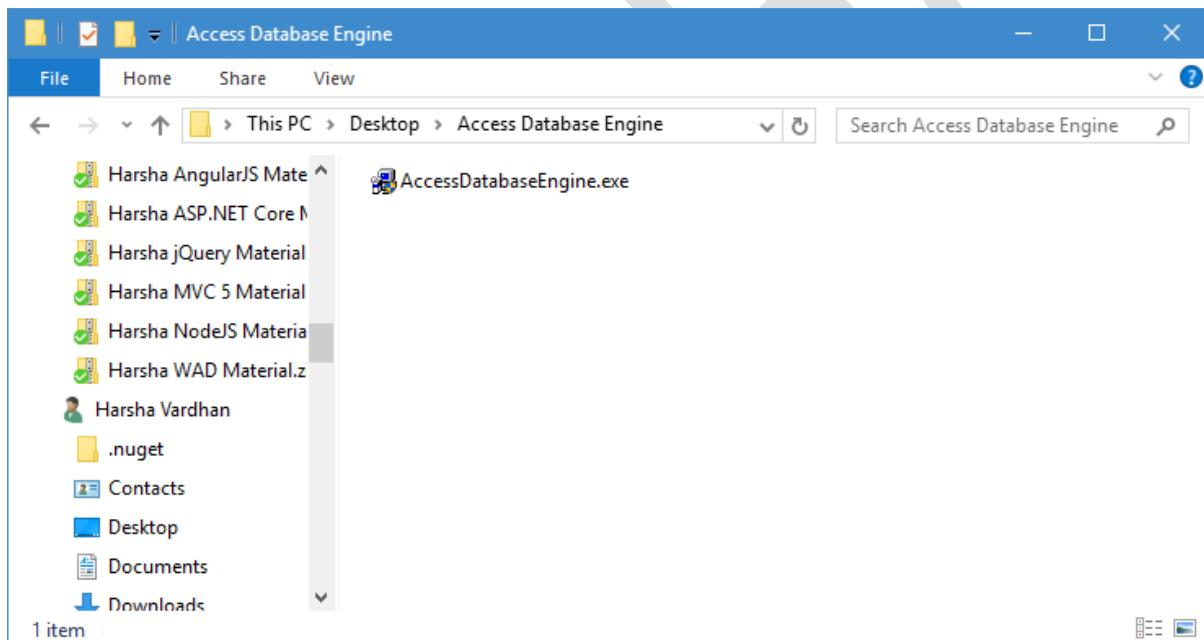
1	scott	5610
2	john	1193
3	allen	1391

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – MS Excel - Example

Installing Access Database Engine

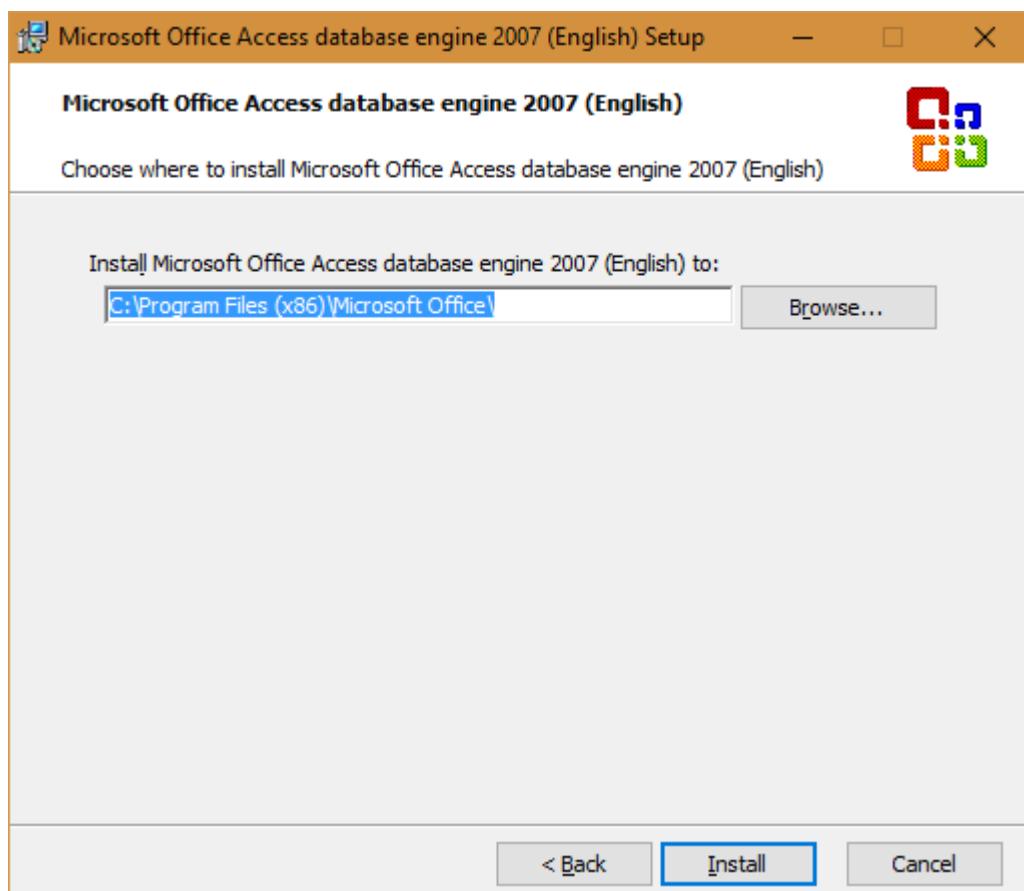
- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at: <https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”.
- Note: Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>
- To install “Access Database Engine”, go to “Access Database Engine” folder.



- Double click on “AccessDatabaseEngine”.
- Click on “Yes”.



- Check the checkbox “I accept the terms in the License Agreement”.
- Click on “Next”.



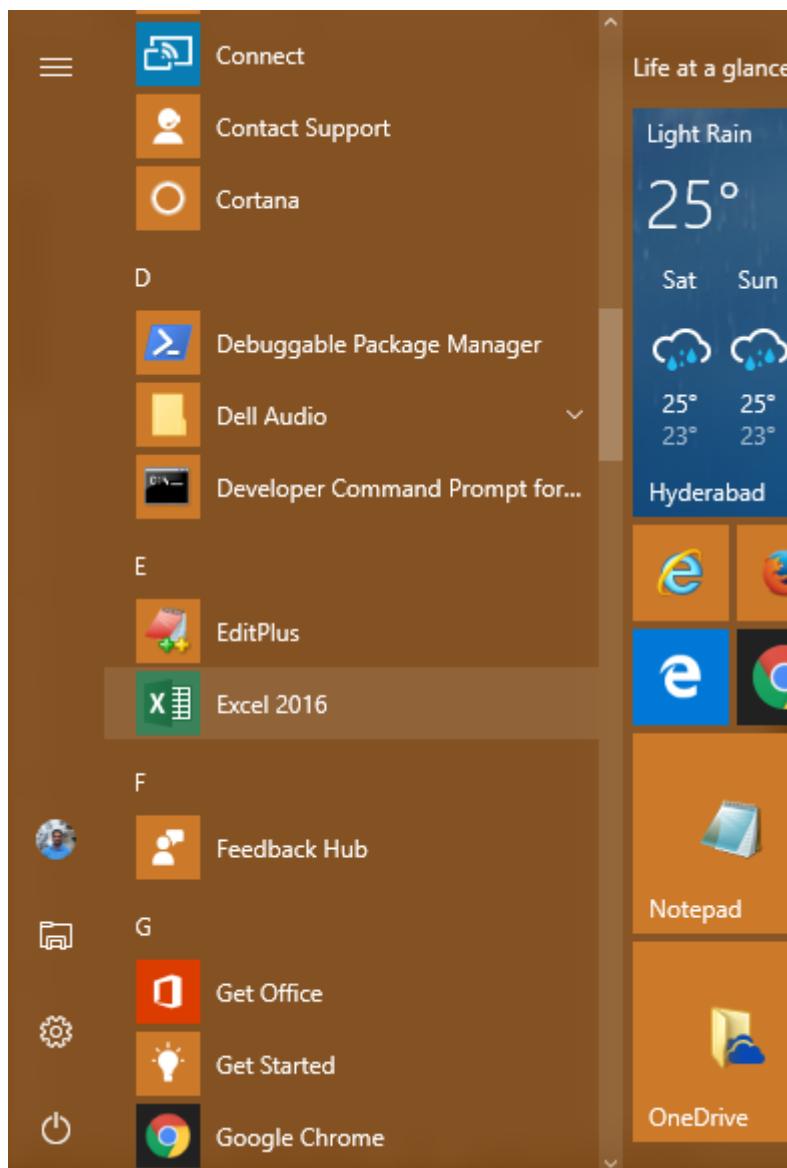
- Click on "Install".
- Installation may take around 1 or 2 minutes.



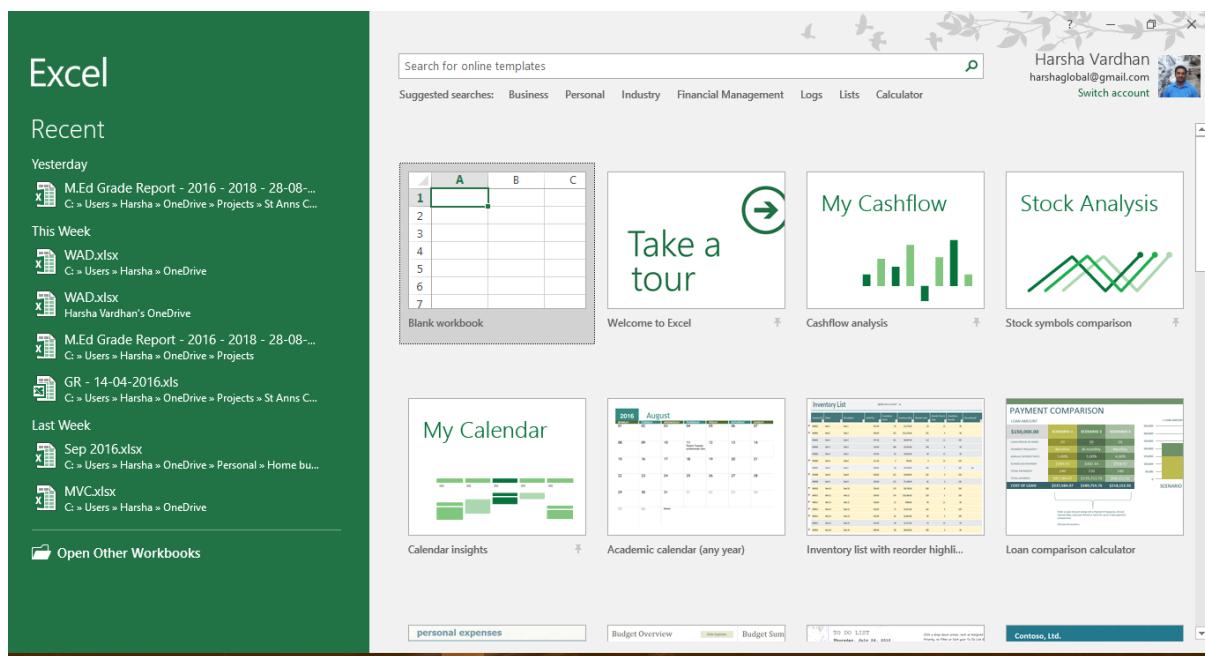
- Click on OK.

Creating work book in MS Excel

- Go to “Start” – “Excel 2016”.



- Click on “Blank Workbook”.

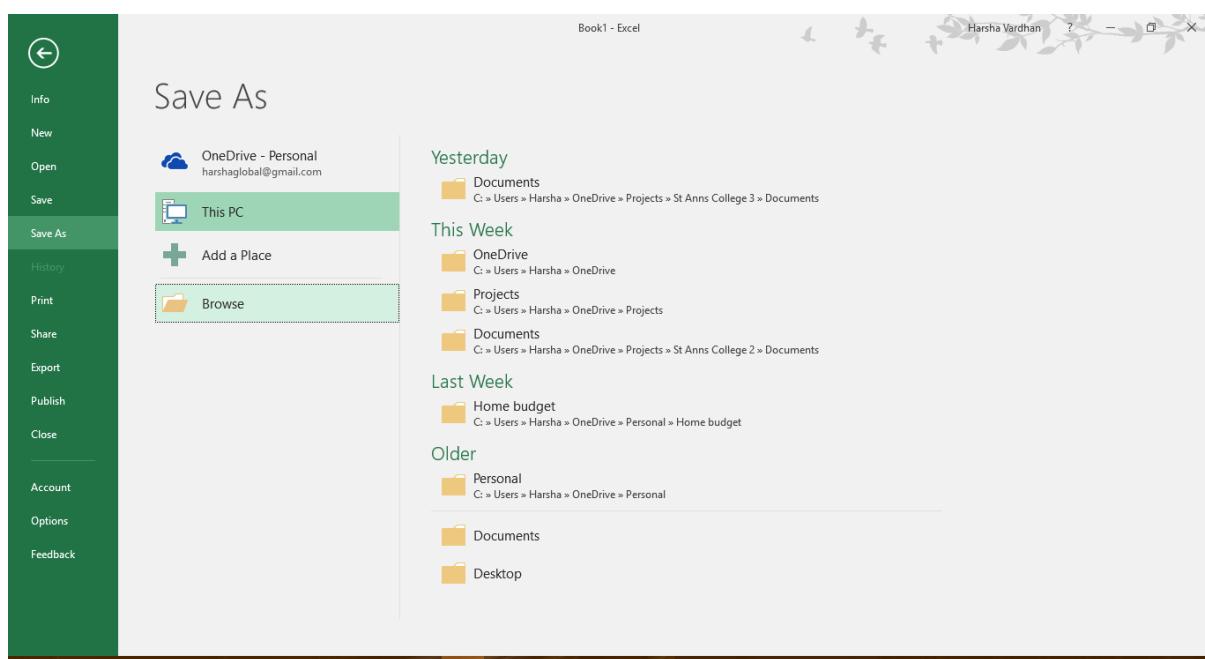


- Type the data as follows:

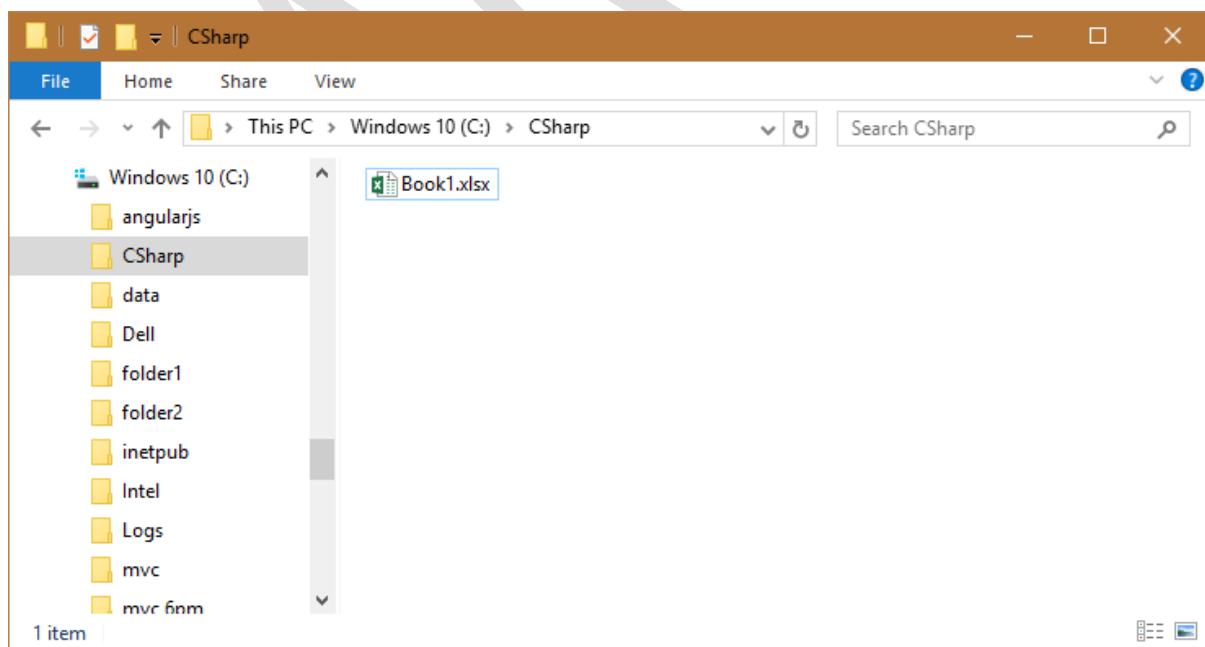
The screenshot shows a Microsoft Excel 2016 spreadsheet titled 'Book1 - Excel'. The table has columns labeled 'EmplID', 'EmpName', and 'Salary'. The data is as follows:

	EmplID	EmpName	Salary
1	1	abc	1000
2	2	def	2000
3	3	ghi	3000
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

- Go to "File" – "Save" – "This PC" – "Browse".



- Select the folder as “C:\CSharp”.
- Type the filename as “Book1.xlsx”.
- Click on “Save”.
- Close “Microsoft Excel 2016”.
- Make sure “Book1.xlsx” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “MSEExcelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “MSEExcelExample”.
- Click on OK. Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.OleDb;
using System.Data;

namespace MSEExcelExample
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "MS Excel";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.AutoScroll = true;
            this.Load += Form1_Load;
        }
    }
}
```

```
}

private void Form1_Load(object sender, EventArgs e)
{
    /* create reference variables */
    OleDbConnection cn;
    OleDbCommand cmd;
    OleDbDataAdapter adp;
    DataSet ds;
    DataTable dt;
    DataRow drow;

    /* create objects */
    cn = new OleDbConnection();
    cmd = new OleDbCommand();
    adp = new OleDbDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\CSharp\Book1.xlsx; Extended Properties='Excel
12.0;HDR=Yes;IMEX=1'";
    cmd.CommandText = "select * from [Sheet1$]";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;

    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];
    int n = 50;
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        drow = dt.Rows[i];

        object obj1, obj2, obj3;
        obj1 = drow["EmpID"];
        obj2 = drow["EmpName"];
```

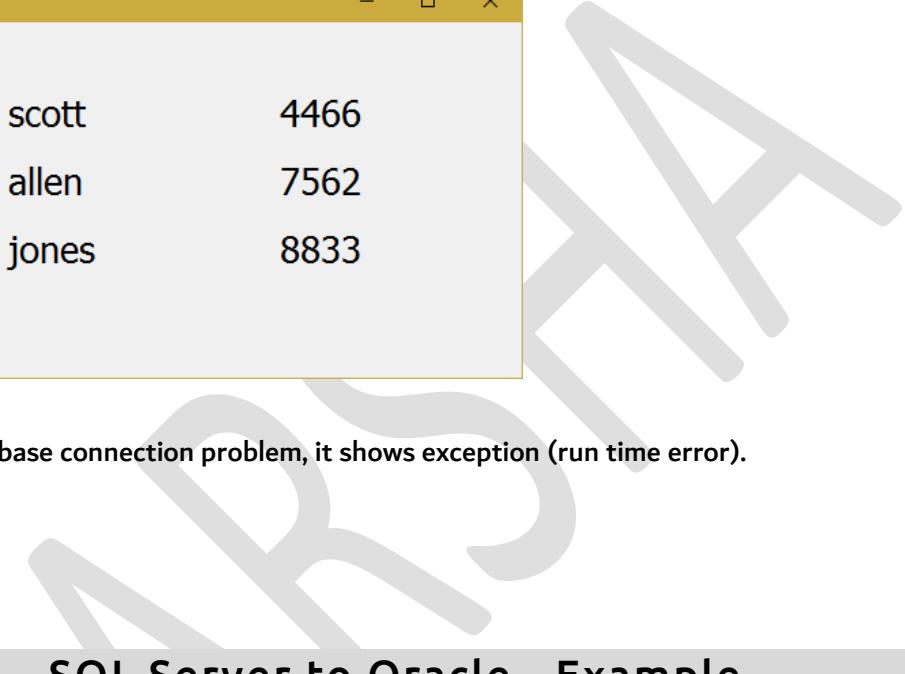
```
obj3 = drow["Salary"];  
  
int eid;  
string ename;  
decimal sal;  
  
eid = Convert.ToInt32(obj1);  
ename = Convert.ToString(obj2);  
sal = Convert.ToDecimal(obj3);  
  
Label label1, label2, label3;  
  
/* label1 */  
label1 = new Label();  
label1.AutoSize = true;  
label1.Text = Convert.ToString(eid);  
label1.Location = new Point(50, n);  
this.Controls.Add(label1);  
  
/* label2 */  
label2 = new Label();  
label2.AutoSize = true;  
label2.Text = ename;  
label2.Location = new Point(150, n);  
this.Controls.Add(label2);  
  
/* label3 */  
label3 = new Label();  
label3.AutoSize = true;  
label3.Text = Convert.ToString(sal);  
label3.Location = new Point(350, n);  
this.Controls.Add(label3);  
  
/* go to next row */  
n += 50;  
}  
}
```

```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output


1	scott	4466
2	allen	7562
3	jones	8833

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – SQL Server to Oracle - Example**Creating SQL Server Database**

- Note: Ignore this step, if you have created “company” database in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

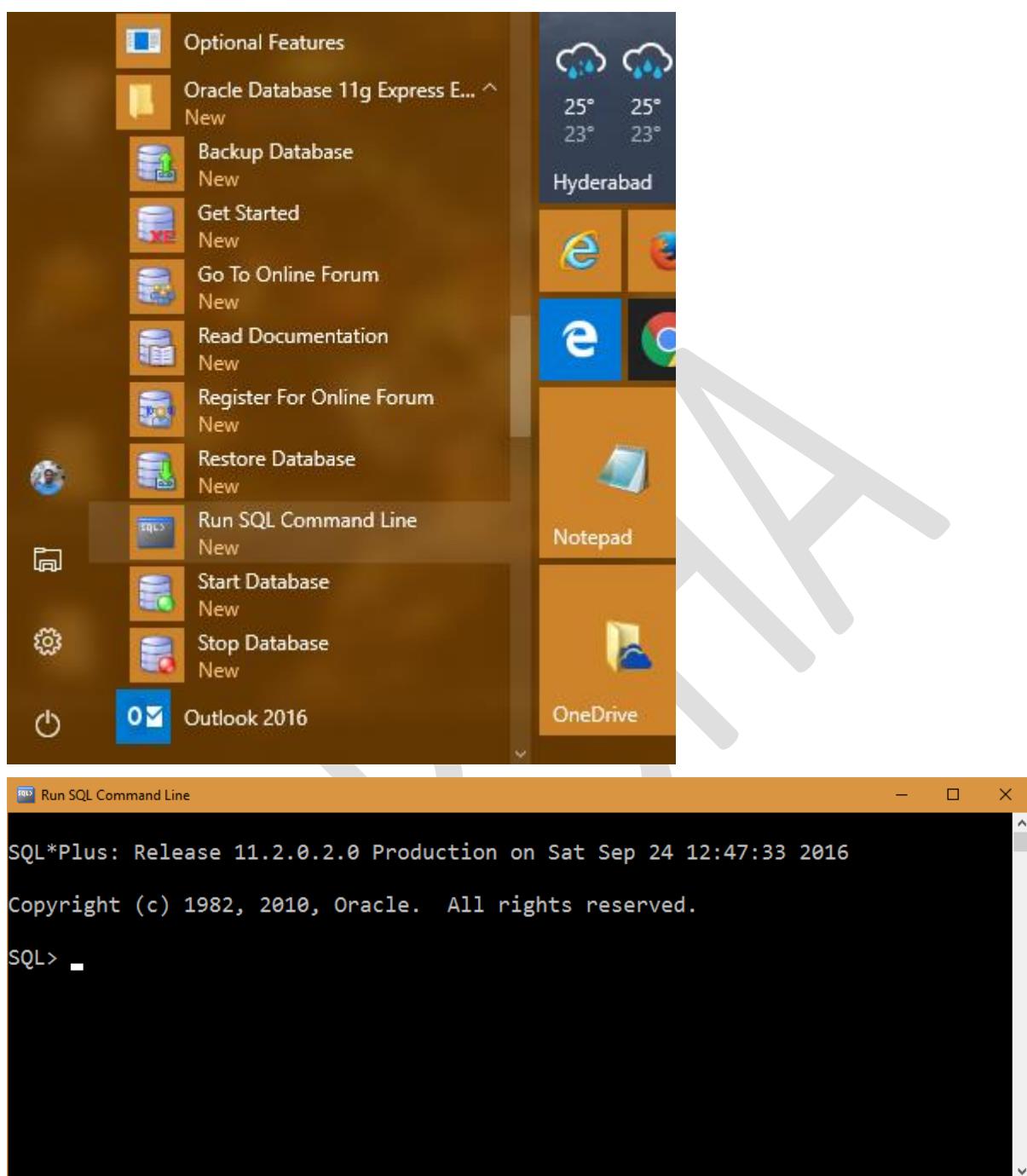
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Installing Oracle Database 11g Expression Edition

- Install “Oracle 11g Express” as shown in the previous example.
- You can download Oracle 11g Express Edition at:
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

Creating table in Oracle

- Note: Ignore this step, if you have created “employees2” table in Oracle already.
- Go to “Start” – “Oracle 11g Express Edition” – “Run SQL Command Line”.



- Type the following script and press Enter.

connect system/123;

```
create table Employees2(  
EmpID int,  
EmpName varchar2(40),  
Salary decimal);
```

commit;



```
Run SQL Command Line
SQL*Plus: Release 11.2.0.2.0 Production on Sat Sep 24 16:02:02 2016
Copyright (c) 1982, 2010, Oracle. All rights reserved.

SQL> connect system/123;
Connected.
SQL>
SQL> create table Employees2(
  2   EmpID int,
  3   EmpName varchar2(40),
  4   Salary decimal);

Table created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> -
```

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SqlServerToOracleExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerToOracleExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
//Copy data from "SqlServer" to "Oracle"
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;

namespace SqlServerToOracleExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "SqlServer to Oracle";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "SqlServer to Oracle";
            button1.AutoSize = true;
            button1.Location = new Point(150, 100);
            button1.Click += button1_Click;
            this.Controls.Add(button1);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            /* create reference variables */
```

```

SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
DataSet ds;
DataTable dt;
DataRow drow;
/* create objects */
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();
/* call properties */
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees2";
cmd.Connection = cn;
adp.SelectCommand = cmd;

/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

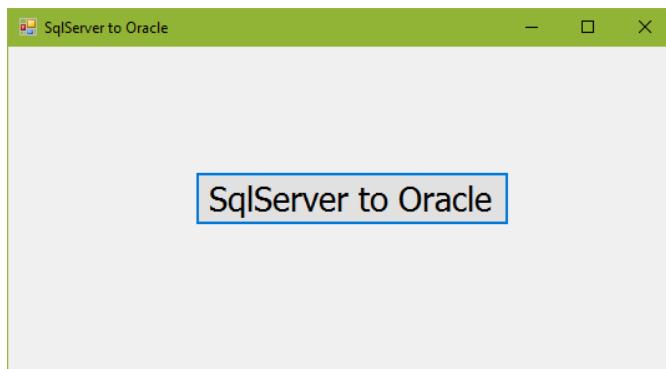
    int eid;
    string ename;
    decimal sal;
    eid = Convert.ToInt32(obj1);
    ename = Convert.ToString(obj2);
    sal = Convert.ToDecimal(obj3);
    InsertIntoOracle(eid, ename, sal);
}

```

```
    MessageBox.Show("Done");
}
private void InsertIntoOracle(int eid, string ename, decimal sal)
{
    //create reference variables
    OleDbConnection cn;
    OleDbCommand cmd;
    //create objects
    cn = new OleDbConnection();
    cmd = new OleDbCommand();
    //calling properties
    cn.ConnectionString = "user id=system; password=123;
provider=msdaora.1";
    cmd.CommandText = string.Format("insert into Employees2
values({0},'{1}',{2})", eid, ename, sal);
    cmd.Connection = cn;
    //calling methods
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Click on “SqlServer to Oracle” button.

Note: If any database connection problem, it shows exception (run time error).

Check the data in “Oracle”:

```
Run SQL Command Line
```

```
SQL> select * from Employees2;

EMPID EMPNAME          SALARY
-----+
      3 Jones            6000
      1 Scott             4000
      2 Allen             5000

SQL>
```

ADO.NET – SQL Server to MS Excel - Example

Creating SQL Server Database

- **Note:** Ignore this step, if you have created “company” database in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database company

go

```
use company
```

```
go
```

```
create table Employees(  

EmpID int primary key,  

EmpName nvarchar(max),  

Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  

insert into Employees values(2, 'Allen', 5000)  

insert into Employees values(3, 'Jones', 6000)  

go
```

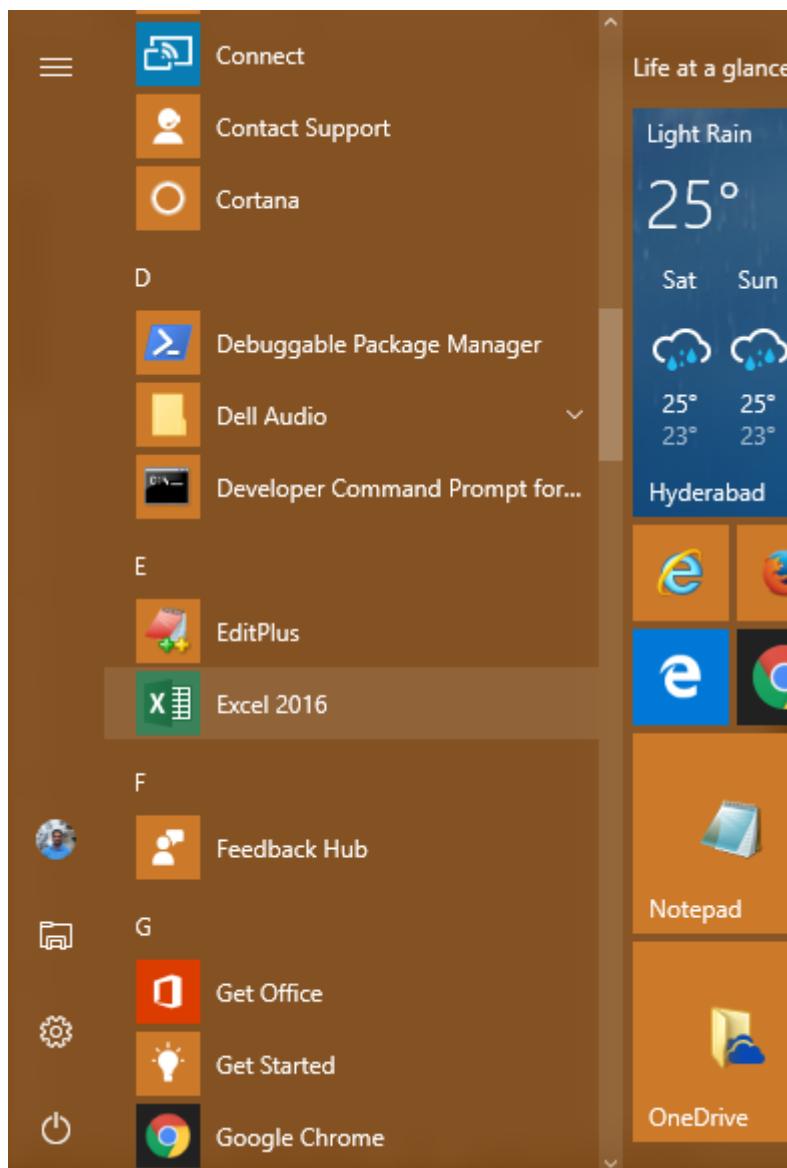
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Installing Access Database Engine

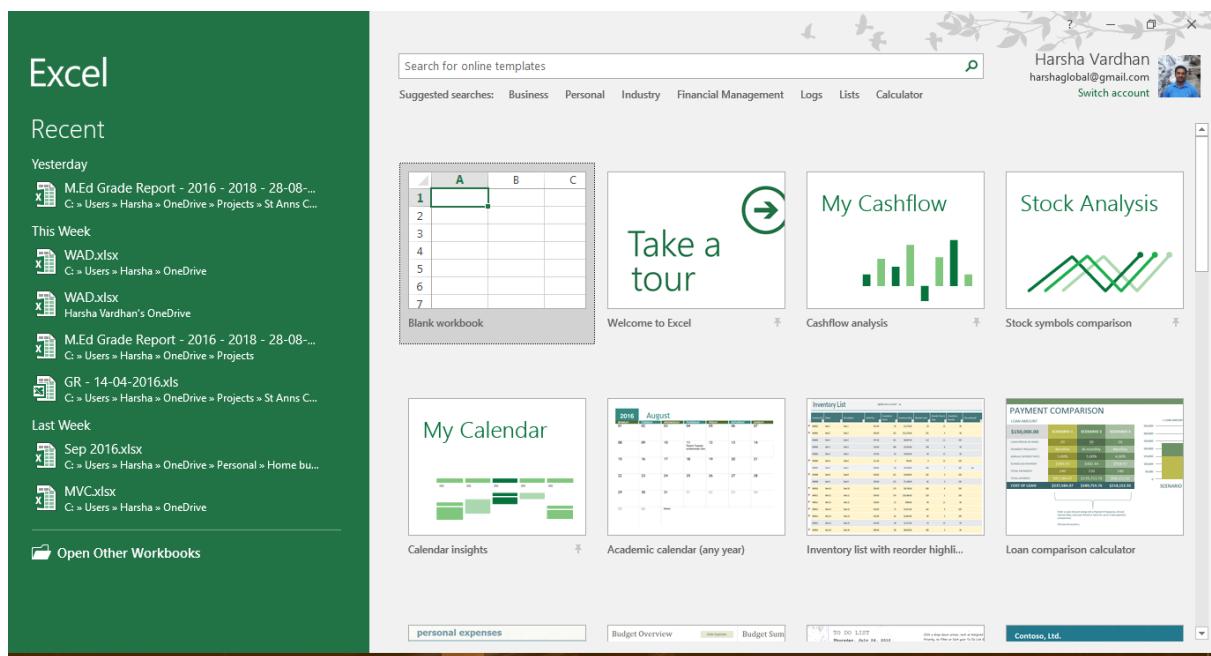
- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at: <https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”, based on the steps explained in the previous examples.
- Note: Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>

Creating work book in MS Excel

- Go to “Start” – “Excel 2016”.



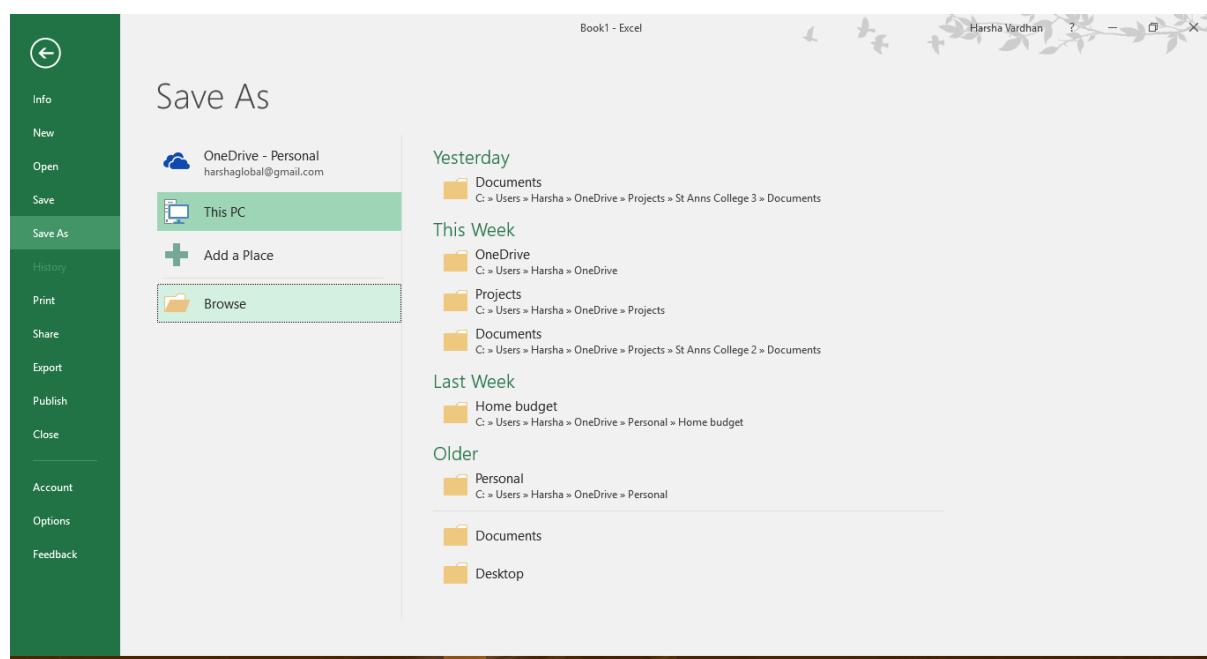
- Click on “Blank Workbook”.



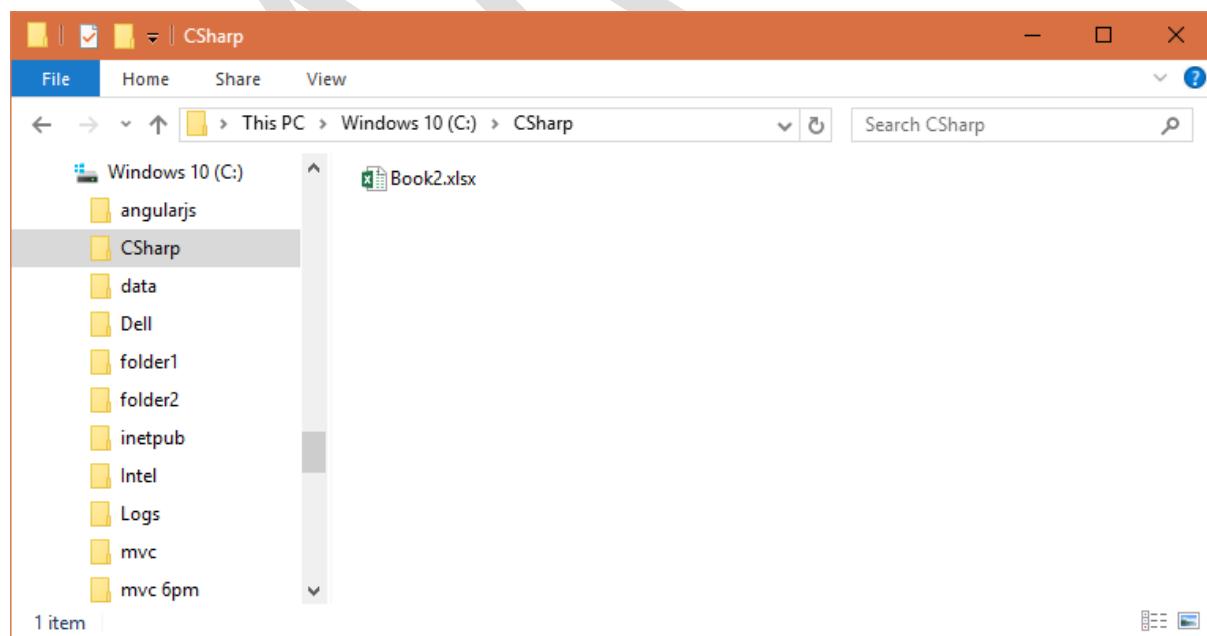
- Type the data as follows:

EmpID	EmpName	Salary
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

- Go to “File” – “Save” – “This PC” – “Browse”.



- Select the folder as “C:\CSharp”.
- Type the filename as “Book2.xlsx”.
- Click on “Save”.
- Close “Microsoft Excel 2016”.
- Make sure “Book2.xlsx” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SqlServerToExcelExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerToExcelExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
//Copy data from "SqlServer" to "Excel"
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;

namespace SqlServerToExcelExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
```

```
this.Text = "SqlServer to Excel";
this.StartPosition = FormStartPosition.CenterScreen;

/* button1 */
button1 = new Button();
button1.Text = "SqlServer to Excel";
button1.AutoSize = true;
button1.Location = new Point(150, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);

}

private void button1_Click(object sender, EventArgs e)
{
    /* create reference variables */
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    DataSet ds;
    DataTable dt;
    DataRow drow;
    /* create objects */
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    ds = new DataSet();
    /* call properties */
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees2";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;
    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];
    for (int i = 0; i < dt.Rows.Count; i++)
    {
```

```

drow = dt.Rows[i];
object obj1, obj2, obj3;
obj1 = drow["EmpID"];
obj2 = drow["EmpName"];
obj3 = drow["Salary"];
int eid;
string ename;
decimal sal;
eid = Convert.ToInt32(obj1);
ename = Convert.ToString(obj2);
sal = Convert.ToDecimal(obj3);

InsertIntoExcel(eid, ename, sal);
}
MessageBox.Show("Done");
}

private void InsertIntoExcel(int eid, string ename, decimal sal)
{
    //create reference variables
    OleDbConnection cn;
    OleDbCommand cmd;

    //create objects
    cn = new OleDbConnection();
    cmd = new OleDbCommand();

    //calling properties
    cn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\CSharp\Book1.xlsx; Extended Properties='Excel
12.0;HDR=Yes;IMEX=3'";
    cmd.CommandText = string.Format("insert into [Sheet1$] values('{0}',
'{1}', '{2}')", eid, ename, sal);
    cmd.Connection = cn;
    //calling methods
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
}

```

```
}
```

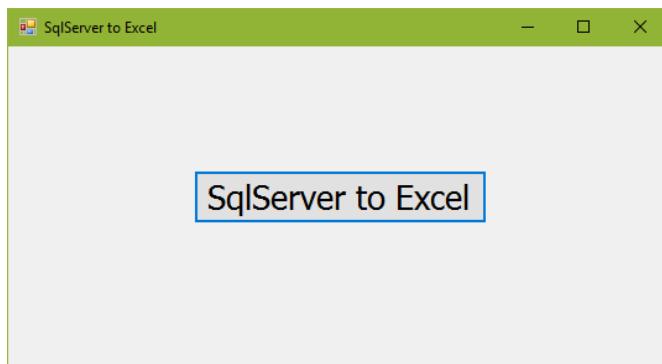
```
}
```

```
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “SqlServer to Excel” button.

Note: If any database connection problem, it shows exception (run time error).

- Check the data in “C:\CSharp\Book2.xlsx” file.



The screenshot shows the Microsoft Excel application interface with the title bar "Book2.xlsx - Excel" and the user name "Harsha Vardhan". The ribbon menu is visible with tabs like File, Home, Insert, Page, Form, Data, Review, View, Add-Ins, Load, Team, Tell me, and Share. The Home tab is selected. The clipboard ribbon group shows "Paste" and other options. The Font ribbon group includes "Font" and "Font Style" dropdowns. The Alignment ribbon group includes "Horizontal", "Vertical", and "Text Orientation" dropdowns. The Number ribbon group includes "Number Format" dropdowns. The Styles ribbon group includes "Conditional Formatting", "Format as Table", "Cell Styles", "Cells", and "Editing" dropdowns. The formula bar shows "C7". The main worksheet area displays a table with columns A, B, and C. Column A contains EmpID values 1, 2, 3, and an empty row 5. Column B contains EmpName values Scott, Allen, Jones, and an empty row 6. Column C contains Salary values 4000, 5000, 6000, and an empty row 7. Row 8 is also empty. The bottom of the screen shows the "Sheet1" tab and various Excel status icons.

	A	B	C	D	E
1	EmpID	EmpName	Salary		
2	1	Scott	4000		
3	2	Allen	5000		
4	3	Jones	6000		
5					
6					
7					
8					

ADO.NET – Oracle to SQL Server - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

go

use company

go

```
create table Employees2(  
EmpID int,  
EmpName nvarchar(max),  
Salary decimal)
```

go

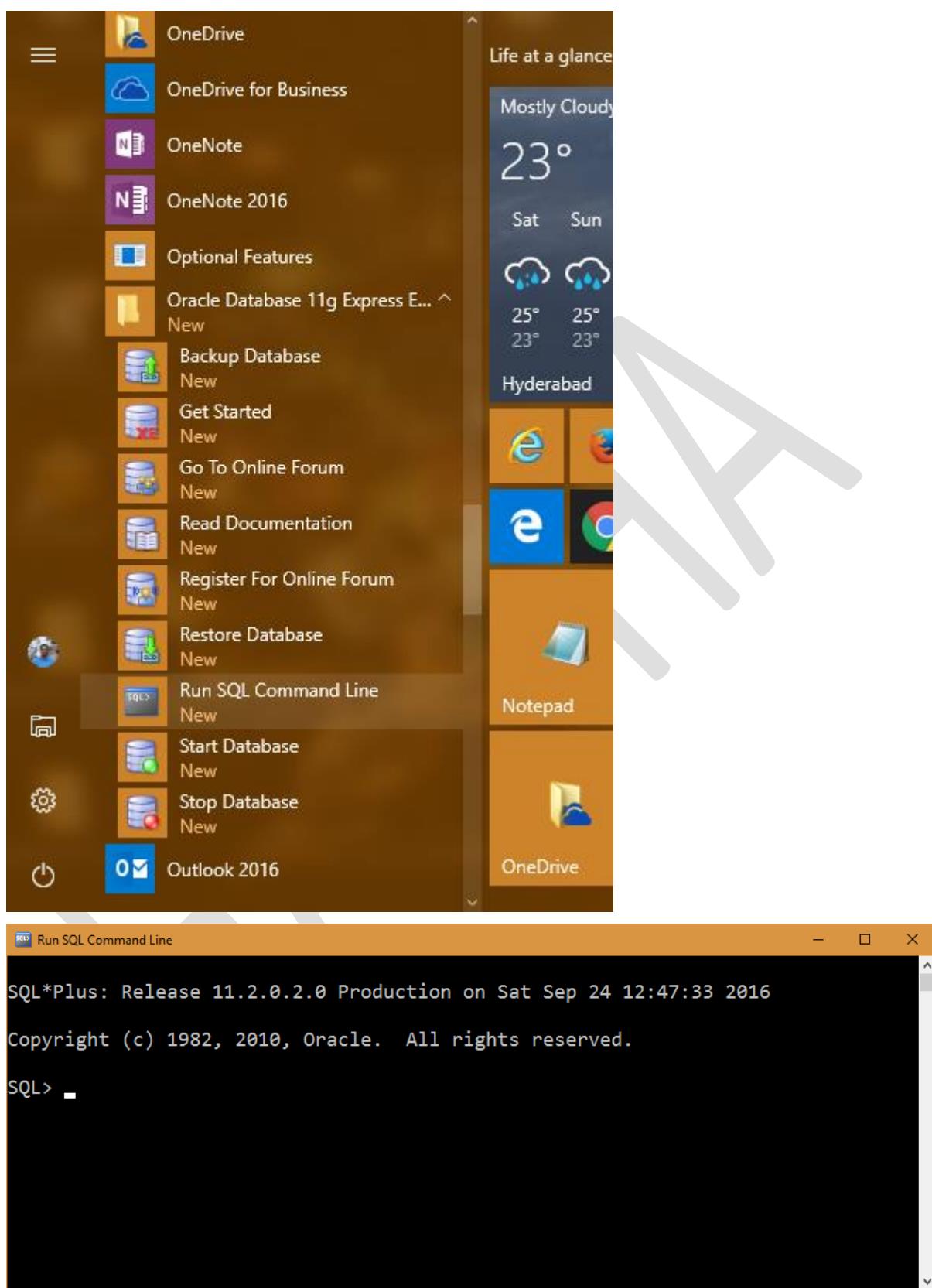
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Installing Oracle Database 11g Expression Edition

- Install “Oracle 11g Express” as shown in the previous example.
- You can download Oracle 11g Express Edition at:
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>

Creating table in Oracle

- Note: Ignore this step, if you have created “employees2” table in Oracle already.
- Go to “Start” – “Oracle 11g Express Edition” – “Run SQL Command Line”.



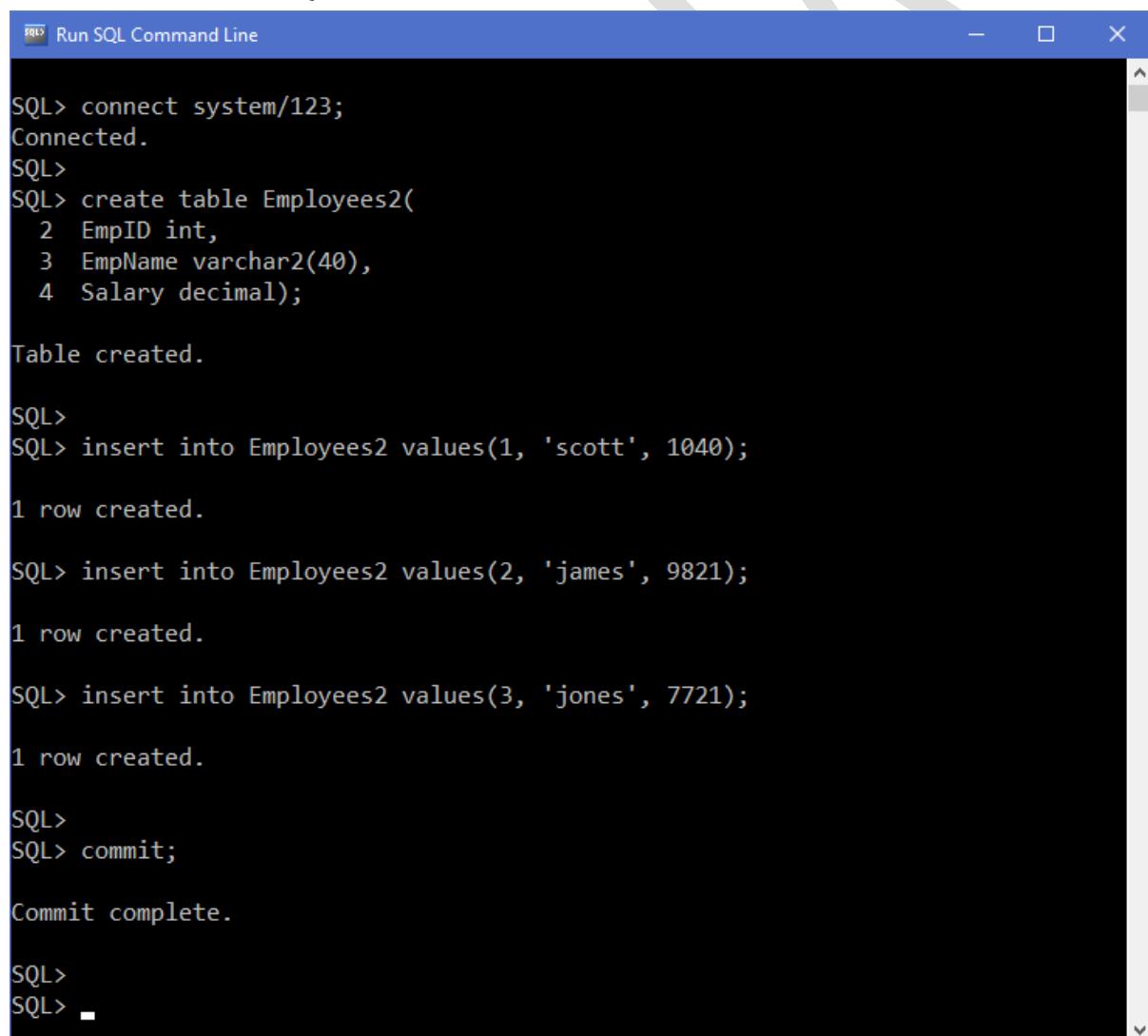
- Type the following script and press Enter.

connect system/123;

```
create table Employees2(
    EmpID int,
    EmpName varchar2(40),
    Salary decimal);

insert into Employees2 values(1, 'scott', 1040);
insert into Employees2 values(2, 'james', 9821);
insert into Employees2 values(3, 'jones', 7721);

commit;
```



The screenshot shows a Windows application window titled "Run SQL Command Line". The window contains the following SQL session:

```
SQL> connect system/123;
Connected.
SQL>
SQL> create table Employees2(
2   EmpID int,
3   EmpName varchar2(40),
4   Salary decimal);

Table created.

SQL>
SQL> insert into Employees2 values(1, 'scott', 1040);

1 row created.

SQL> insert into Employees2 values(2, 'james', 9821);

1 row created.

SQL> insert into Employees2 values(3, 'jones', 7721);

1 row created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> -
```

Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “OracleToSqlServerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “OracleToSqlServerExample”. Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
//Copy data from "Oracle" to "SqlServer"
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;

namespace OracleToSqlServerExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "Oracle to SqlServer";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Oracle to SqlServer";
```

```
button1.AutoSize = true;
button1.Location = new Point(150, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);
}

private void button1_Click(object sender, EventArgs e)
{
    /* create reference variables */
    OleDbConnection cn;
    OleDbCommand cmd;
    OleDbDataAdapter adp;
    DataSet ds;
    DataTable dt;
    DataRow drow;

    /* create objects */
    cn = new OleDbConnection();
    cmd = new OleDbCommand();
    adp = new OleDbDataAdapter();
    ds = new DataSet();

    /* call properties */
    cn.ConnectionString = "user id=system; password=123;
provider=msdaora.1";
    cmd.CommandText = "select * from Employees2";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;
    /* call methods */
    adp.Fill(ds);
    dt = ds.Tables[0];
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        drow = dt.Rows[i];
        object obj1, obj2, obj3;
        obj1 = drow["EmpID"];
        obj2 = drow["EmpName"];
```

```

obj3 = drow["Salary"];

int eid;
string ename;
decimal sal;
eid = Convert.ToInt32(obj1);
ename = Convert.ToString(obj2);
sal = Convert.ToDecimal(obj3);
InsertIntoSqlServer(eid, ename, sal);
}

MessageBox.Show("Done");
}

private void InsertIntoSqlServer(int eid, string ename, decimal sal)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    //calling properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = string.Format("insert into Employees2
values({0},'{1}',{2})", eid, ename, sal);
    cmd.Connection = cn;
    //calling methods
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
}
}
}

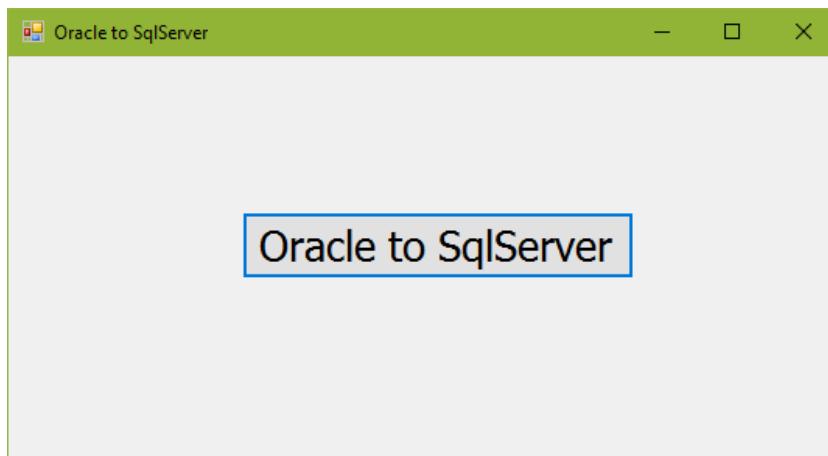
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

HARSHA

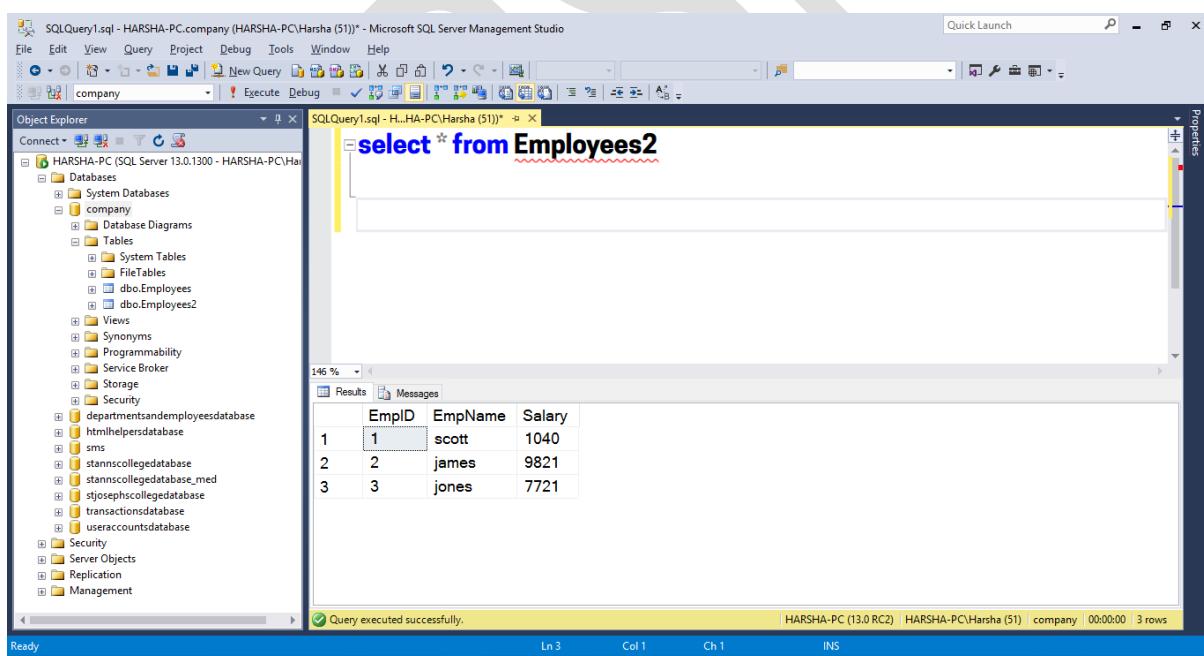
Output



Click on "Oracle to SqlServer" button.

Note: If any database connection problem, it shows exception (run time error).

Check the data in "SQL Server":



ADO.NET – Excel to SQL Server - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees2(
```

```
EmpID int,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

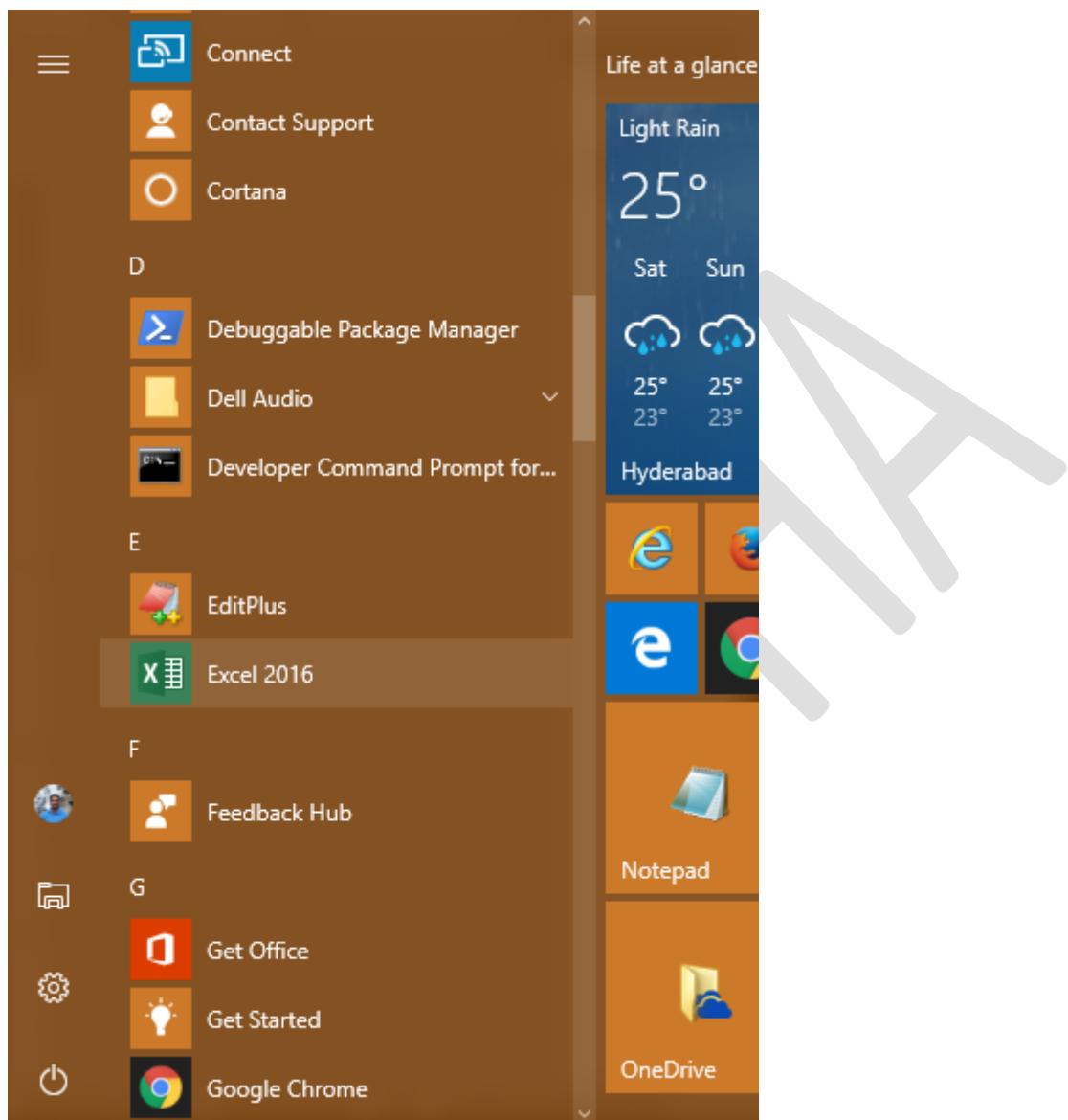
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Installing Access Database Engine

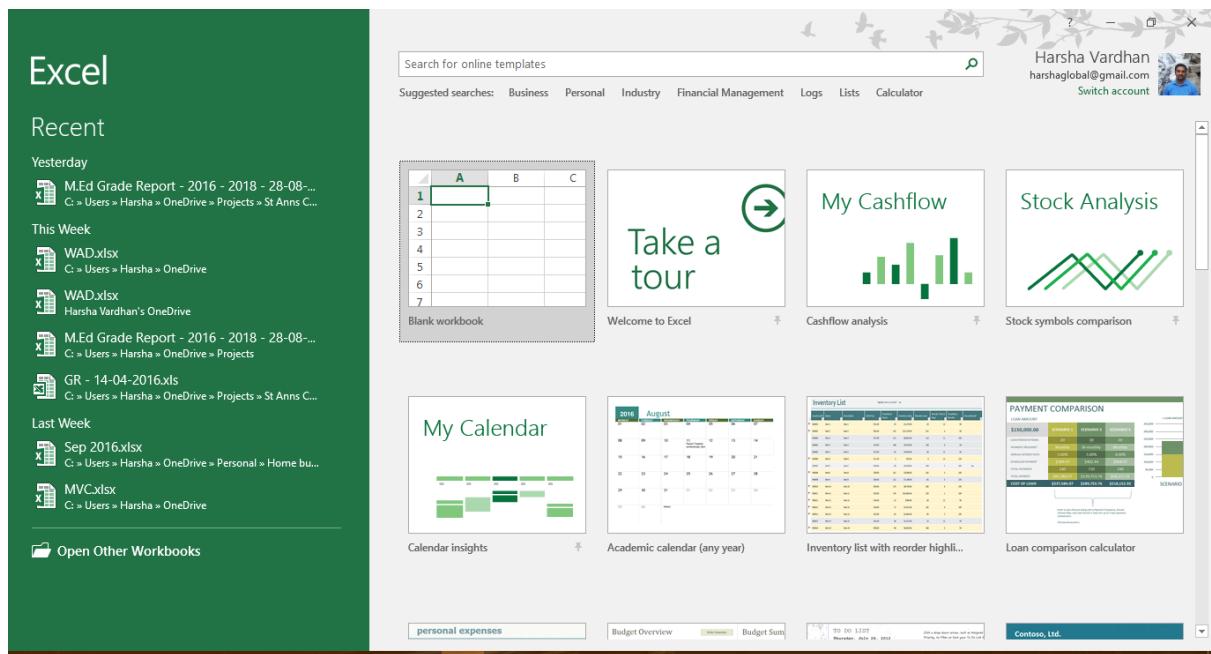
- Install MS Office 2007 or 2010 or 2013 or 2016. You can download MS Office 2016 at:
<https://support.office.com/en-us/article/Download-and-install-or-reinstall-Office-365-Office-2016-or-Office-2013-on-your-PC-or-Mac-4414eaaf-0478-48be-9c42-23adc4716658?ui=en-US&rs=en-US&ad=US>
- Additionally, we have to install “Access Database Engine”, based on the steps explained in the previous examples.
- Note: Ignore this step, if you have installed “Access Database Engine” already.
- You can download “MS Access Database Engine” at: <https://www.microsoft.com/en-in/download/details.aspx?id=13255>

Creating Workbook in “Excel”:

- Go to “Start” – “Excel 2016”.



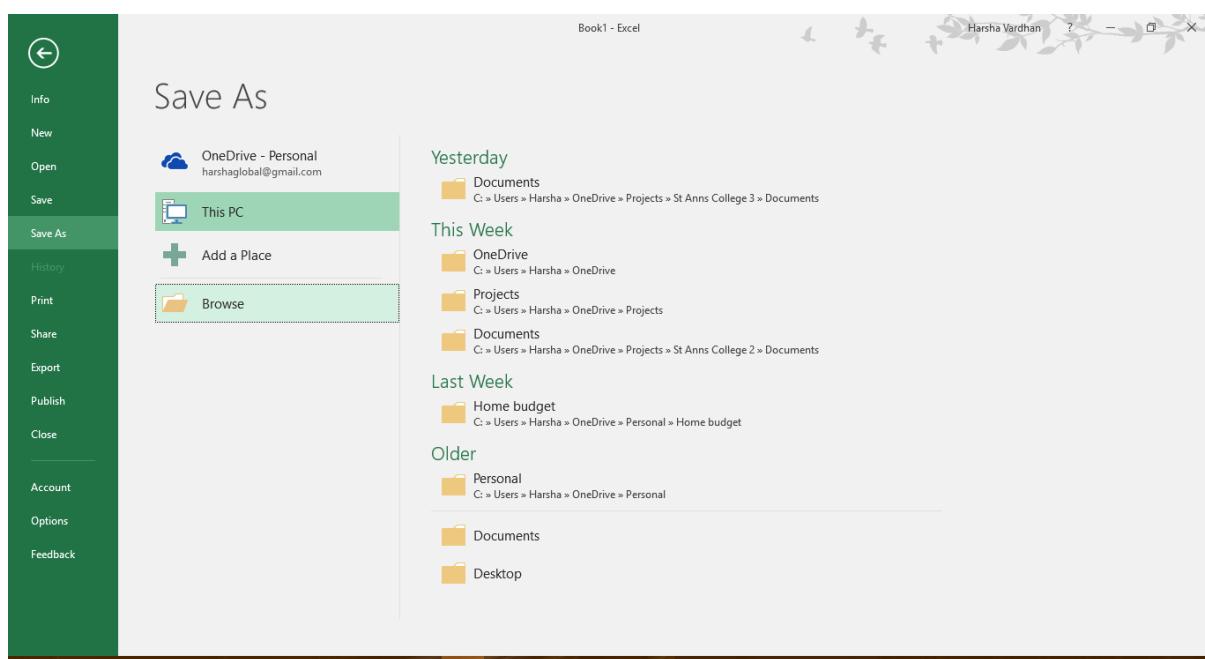
- Click on “Blank Workbook”.



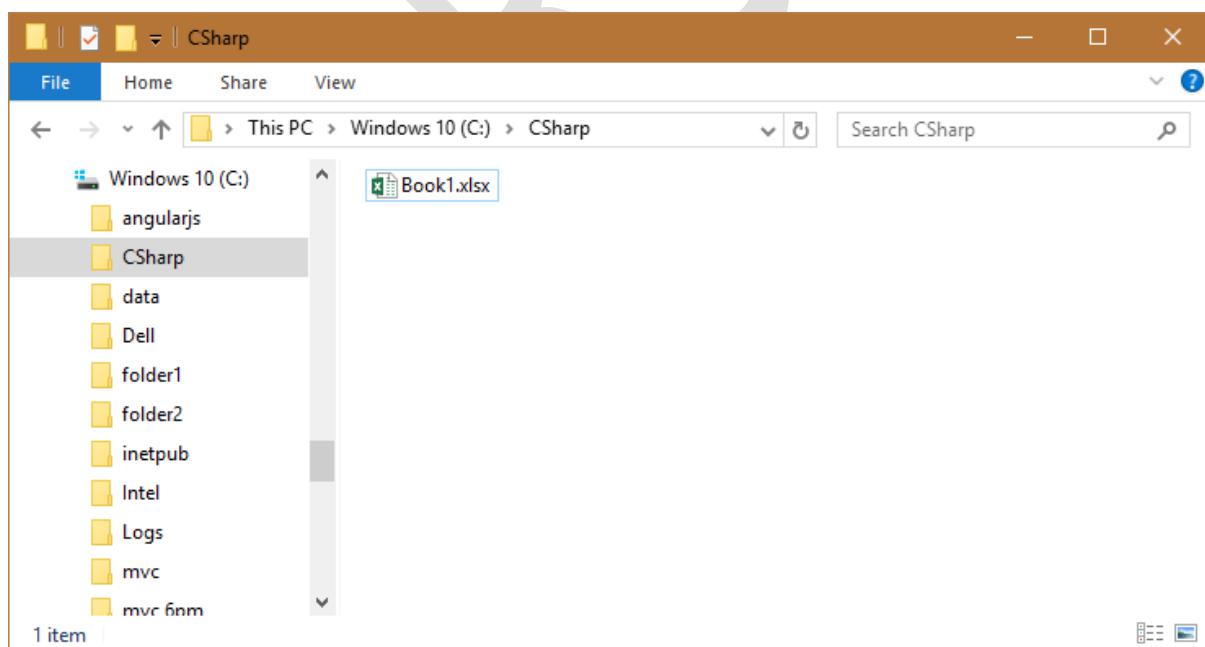
- Type the data as follows:

Book1 - Excel													
File Home Insert Page Layout Formulas Data Review View Add-ins Load Test Team Tell me what you want to do Harsha Vardhan Share													
Clipboard Font Alignment Number Conditional Formatting Styles Cells Editing													
1	EmplID	EmpName	Salary										
2	1	abc	1000										
3	2	def	2000										
4	3	ghi	3000										
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
Sheet1													

- Go to “File” – “Save” – “This PC” – “Browse”.



- Select the folder as “C:\CSharp”.
- Type the filename as “Book1.xlsx”.
- Click on “Save”. Close “Microsoft Excel 2016”.
- Make sure “Book1.xlsx” file exist in “C:\CSharp” folder.



Creating Project

- Open Visual Studio 2019.

- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “ExcelToSqlServerExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “ExcelToSqlServerExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
//Copy data from "Excel" to "SqlServer"
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Data;

namespace ExcelToSqlServerExample
{
    public partial class Form1 : Form
    {
        Button button1;
        public Form1()
        {
            InitializeComponent();
            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "Excel to SqlServer";
            this.StartPosition = FormStartPosition.CenterScreen;
            /* button1 */
            button1 = new Button();
        }
    }
}
```

```

button1.Text = "Excel to SqlServer";
button1.AutoSize = true;
button1.Location = new Point(150, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);
}
private void button1_Click(object sender, EventArgs e)
{
/* create reference variables */
OleDbConnection cn;
OleDbCommand cmd;
OleDbDataAdapter adp;
DataSet ds;
DataTable dt;
DataRow drow;
/* create objects */
cn = new OleDbConnection();
cmd = new OleDbCommand();
adp = new OleDbDataAdapter();
ds = new DataSet();
/* call properties */
cn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data
Source=C:\CSharp\Book1.xlsx; Extended Properties='Excel
12.0;HDR=Yes;IMEX=1'";
cmd.CommandText = "select * from [Sheet1$]";
cmd.Connection = cn;
adp.SelectCommand = cmd;
/* call methods */
adp.Fill(ds);
dt = ds.Tables[0];
for (int i = 0; i < dt.Rows.Count; i++)
{
    drow = dt.Rows[i];
    object obj1, obj2, obj3;
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];
}

```

```

int eid;
string ename;
decimal sal;
eid = Convert.ToInt32(obj1);
ename = Convert.ToString(obj2);
sal = Convert.ToDecimal(obj3);
InsertIntoSqlServer(eid, ename, sal);
}
MessageBox.Show("Done");
}

private void InsertIntoSqlServer(int eid, string ename, decimal sal)
{
//create reference variables
SqlConnection cn;
SqlCommand cmd;
//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = string.Format("insert into Employees2
values({0}, '{1}', {2})", eid, ename, sal);
cmd.Connection = cn;
cn.Open();
cmd.ExecuteNonQuery();
cn.Close();
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

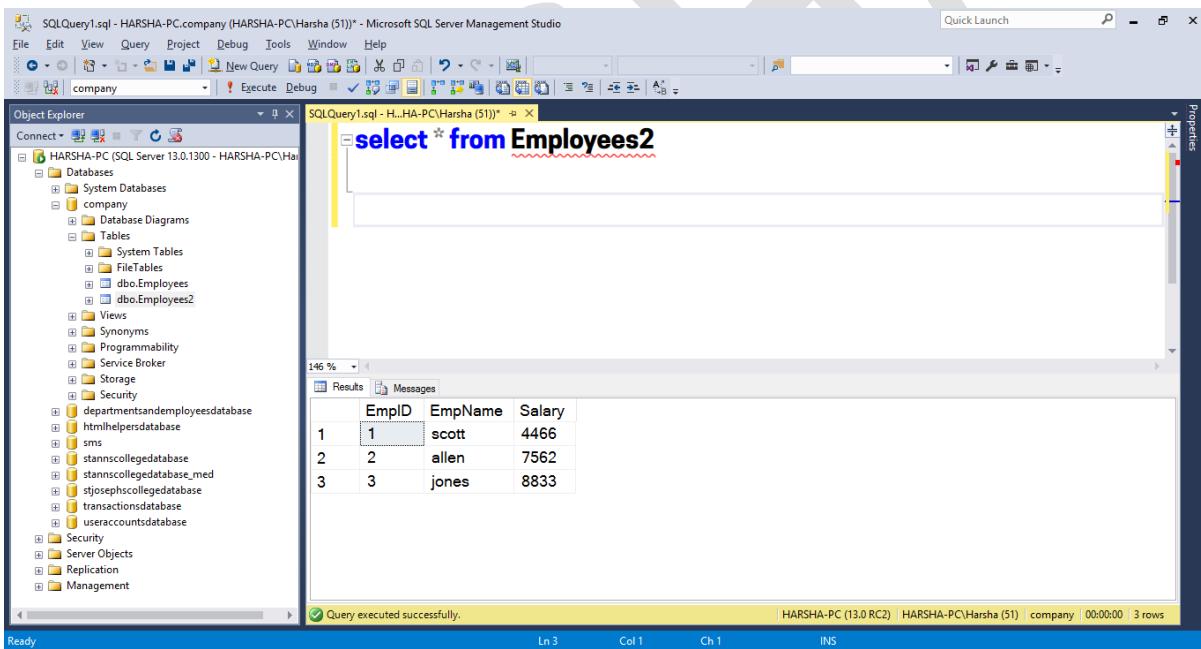
Output



Click on “Excel to SqlServer” button.

Note: If any database connection problem, it shows exception (run time error).

Check the data in “SQL Server”:



ADO.NET – SQL Server to File - Example

Creating SQL Server Database

- **Note:** Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.

- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees2(
```

```
EmpID int,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “SqlServerToFileExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “SqlServerToFileExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
//Copy data from "SqlServer" to "File"
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;
using System.Diagnostics;
using System.IO;

namespace SqlServerToFileExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "SqlServer to File";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "SqlServer to File";
            button1.AutoSize = true;
            button1.Location = new Point(150, 100);
            button1.Click += button1_Click;
            this.Controls.Add(button1);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //create directory if not exists
```

```
if (Directory.Exists(@"C:\CSharp") == false)
{
    Directory.CreateDirectory(@"C:\CSharp");
}

//create reference variables
SqlConnection cn;
SqlCommand cmd;
SqlDataAdapter adp;
DataSet ds;
DataTable dt;
DataRow drow;
FileInfo finfo;
FileStream fs;
StreamWriter sw;

//create objects
cn = new SqlConnection();
cmd = new SqlCommand();
adp = new SqlDataAdapter();
ds = new DataSet();
finfo = new FileInfo(@"C:\CSharp\Employees.txt");

//delete the file if already exists
if (finfo.Exists == true)
    finfo.Delete();

//create objects
fs = new FileStream(@"C:\CSharp\Employees.txt", FileMode.Create,
FileAccess.Write);
sw = new StreamWriter(fs);

//call properties
cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
cmd.CommandText = "select * from Employees2";
cmd.Connection = cn;
```

```

adp.SelectCommand = cmd;

//call methods
adp.Fill(ds);

//read data
dt = ds.Tables[0];

//read data from database record-by-record
for (int i = 0; i < dt.Rows.Count; i++)
{
    //get values
    object obj1, obj2, obj3;
    drow = dt.Rows[i];
    obj1 = drow["EmpID"];
    obj2 = drow["EmpName"];
    obj3 = drow["Salary"];

    int eid = Convert.ToInt32(obj1);
    string ename = Convert.ToString(obj2);
    double sal = Convert.ToDouble(obj3);
    string msg = eid + "," + ename + "," + sal;
    //write data to the file
    sw.WriteLine(msg);
}

//close the file
sw.Close();

MessageBox.Show("Data has been written to the following file
successfully.\nC:\\\\CSharp\\\\Employees.txt");

//open the file in notepad
Process.Start(@"C:\\Windows\\System32\\notepad.exe",
@"C:\\CSharp\\Employees.txt");
}
}

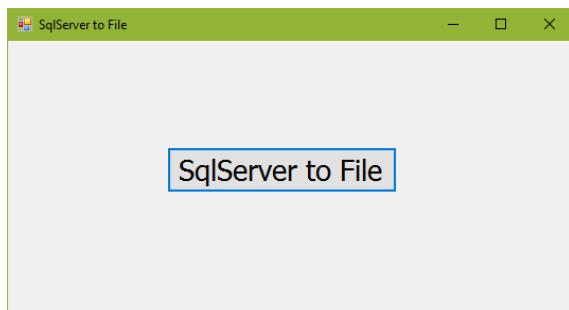
```

}

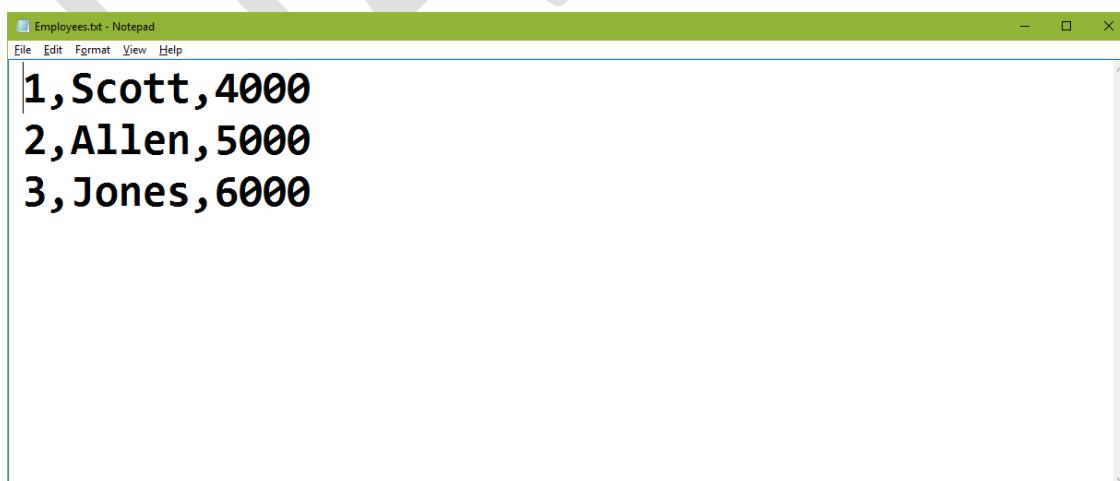
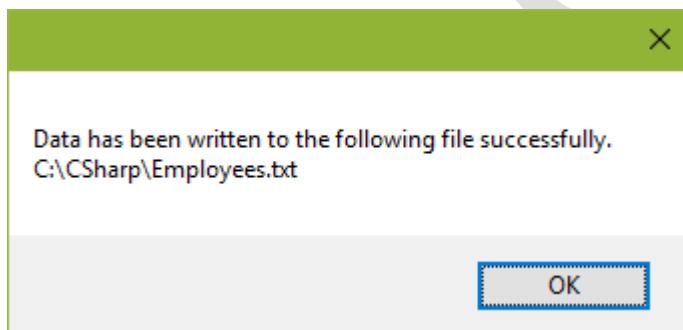
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “SqlServer to File”.



Note: If any database connection problem, it shows exception (run time error).

ADO.NET – File to SQL Server - Example

Creating SQL Server Database

- Note: Ignore this step, if you have created “company” database with “Employees2” table in SQL Server, already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees2(
```

```
EmpID int,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “FileToSqlServerExample”.
- Type the location as “C:\CSharp”.

- Type the solution name as “FileToSqlServerExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
//Copy data from "File" to "SqlServer"
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.IO;

namespace FileToSqlServerExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(550, 300);
            this.Text = "File to SqlServer";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "File to SqlServer";
            button1.AutoSize = true;
            button1.Location = new Point(150, 100);
            button1.Click += button1_Click;
            this.Controls.Add(button1);
        }
    }
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    FileInfo finfo;
    FileStream fs;
    StreamReader sr;
    SqlParameter p1, p2, p3;
    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    p1 = new SqlParameter();
    p2 = new SqlParameter();
    p3 = new SqlParameter();
    finfo = new FileInfo(@"C:\CSharp\Employees.txt");
    //check whether the file already exists or not
    if (finfo.Exists == false)
    {
        MessageBox.Show("File not exists.");
        return;
    }
    //create objects
    fs = new FileStream(@"C:\CSharp\Employees.txt", FileMode.Open,
    FileAccess.Read);
    sr = new StreamReader(fs);
    //call properties
    cn.ConnectionString = "data source=localhost; integrated
    security=yes; initial catalog=company";
    cmd.CommandText = "insert into Employees2 values(@empid,
    @empname, @salary)";
    cmd.Connection = cn;
    //call methods
    cmd.Parameters.Add(p1);
    cmd.Parameters.Add(p2);
    cmd.Parameters.Add(p3);
```

```

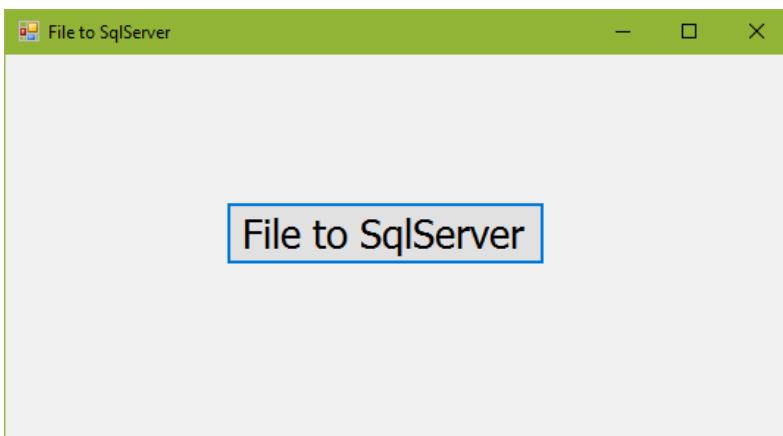
//read all lines from the file
while (true)
{
    //get a line from the file
    string line = sr.ReadLine();
    string[] values;
    values = line.Split(',');
    int eid = Convert.ToInt32(values[0]);
    string ename = Convert.ToString(values[1]);
    double sal = Convert.ToDouble(values[2]);
    //set data into parameters
    p1.ParameterName = "@empid";
    p1.Value = eid;
    p2.ParameterName = "@empname";
    p2.Value = ename;
    p3.ParameterName = "@salary";
    p3.Value = sal;
    //insert into database
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
    if (sr.EndOfStream == true)
        break;
}
//close the file
sr.Close();
MessageBox.Show("Data stored into database");
}
}
}

```

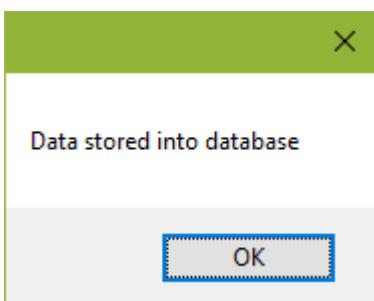
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Click on “File to SqlServer”.



Note: If any database connection problem, it shows exception (run time error).

Check the data in “SQL Server”:

EmplID	EmpName	Salary
1	Scott	4000
2	Allen	5000
3	Jones	6000

HARSHA

ADO.NET – SqlCommandBuilder – DataSet – Insertion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.

- Type the project name as “DataSetInsertionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetInsertionExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DataSetInsertionExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "DataSet - Insertion";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Emp ID: ";
        }
    }
}
```

```
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* label2 */
label2 = new Label();
label2.AutoSize = true;
label2.Text = "Emp Name: ";
label2.Location = new Point(50, 100);
this.Controls.Add(label2);

/* label3 */
label3 = new Label();
label3.AutoSize = true;
label3.Text = "Salary:";
label3.Location = new Point(50, 150);
this.Controls.Add(label3);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(300, 50);
textbox1.Location = new Point(250, 50);
this.Controls.Add(textbox1);

/* textbox2 */
textbox2 = new TextBox();
textbox2.Size = new Size(300, 50);
textbox2.Location = new Point(250, 100);
this.Controls.Add(textbox2);

/* textbox3 */
textbox3 = new TextBox();
textbox3.Size = new Size(300, 50);
textbox3.Location = new Point(250, 150);
this.Controls.Add(textbox3);

/* button1 */
button1 = new Button();
```

```
button1.AutoSize = true;
button1.Text = "Insert";
button1.Location = new Point(250, 200);
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);

}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlCommandBuilder cmb;
    DataSet ds;
    DataTable dt;
    DataRow drow;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    cmb = new SqlCommandBuilder();
    ds = new DataSet();

    //call properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;
    cmbDataAdapter = adp;

    //call methods
    adp.Fill(ds);
    dt = ds.Tables[0];
```

```

//create a new data row
drow = dt.NewRow();
drow["empid"] = textBox1.Text;
drow["empname"] = textBox2.Text;
drow["salary"] = textBox3.Text;

//add row to the table
dt.Rows.Add(drow);

//save changes back to the database
adp.Update(ds);

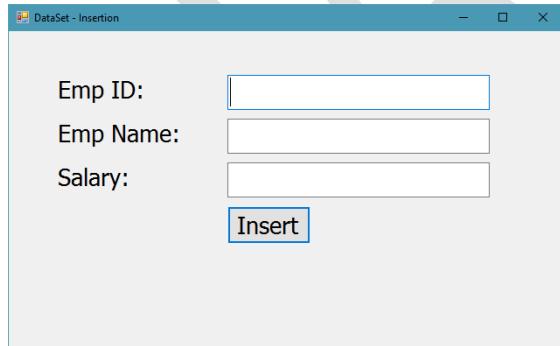
MessageBox.Show("Inserted");
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some emp id, emp name and salary and click on “Insert”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – SqlCommandBuilder – DataSet – Updation - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.

- Select “Windows Forms Application”.
- Type the project name as “DataSetUpdationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetUpdationExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DataSetUpdationExample
{
    public partial class Form1 : Form
    {
        Label label1, label2, label3;
        TextBox textbox1, textbox2, textbox3;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 400);
            this.Text = "DataSet - Updation";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
```

```
label1.Text = "Existing Emp ID:";  
label1.Location = new Point(50, 50);  
this.Controls.Add(label1);  
  
/* label2 */  
label2 = new Label();  
label2.AutoSize = true;  
label2.Text = "Emp Name:";  
label2.Location = new Point(50, 100);  
this.Controls.Add(label2);  
  
/* label3 */  
label3 = new Label();  
label3.AutoSize = true;  
label3.Text = "Salary:";  
label3.Location = new Point(50, 150);  
this.Controls.Add(label3);  
  
/* textbox1 */  
textbox1 = new TextBox();  
textbox1.Size = new Size(300, 50);  
textbox1.Location = new Point(300, 50);  
this.Controls.Add(textbox1);  
  
/* textbox2 */  
textbox2 = new TextBox();  
textbox2.Size = new Size(300, 50);  
textbox2.Location = new Point(300, 100);  
this.Controls.Add(textbox2);  
  
/* textbox3 */  
textbox3 = new TextBox();  
textbox3.Size = new Size(300, 50);  
textbox3.Location = new Point(300, 150);  
this.Controls.Add(textbox3);  
  
/* button1 */
```

```
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Update";
button1.Location = new Point(300, 200);
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);

}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlCommandBuilder cmb;
    DataSet ds;
    DataTable dt;
    DataRow drow;

    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    cmb = new SqlCommandBuilder();
    ds = new DataSet();

    //call properties
    cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
    cmd.CommandText = "select * from Employees";
    cmd.Connection = cn;
    adp.SelectCommand = cmd;
    cmbDataAdapter = adp;

    //call methods
    adp.Fill(ds);
}
```

```

dt = ds.Tables[0];

//get datarow based on empid
int n = Convert.ToInt32(textBox1.Text);
string condition = "empid=" + n;
drow = dt.Select(condition)[0];

//update
drow["empname"] = textBox2.Text;
drow["salary"] = textBox3.Text;

//save changes back to the database
adp.Update(ds);

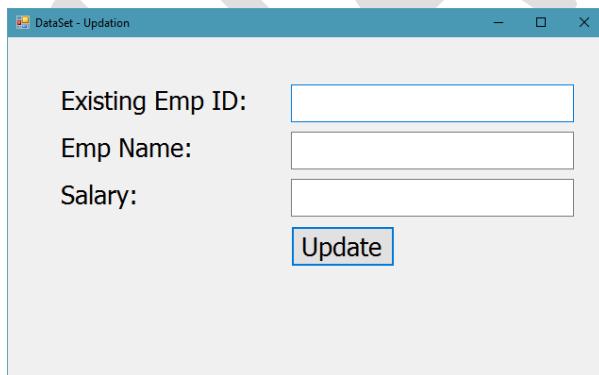
MessageBox.Show("Updated");
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Type some existing emp id, new emp name and new salary and click on “Update”.

Note: If any database connection problem, it shows exception (run time error).

ADO.NET – SqlCommandBuilder – DataSet – Deletion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.

- Type the project name as “DataSetDeletionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “DataSetDeletionExample”.
- Click on OK.
- Right click on Form1 and click on “View Code”.

Form1.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Data.SqlClient;
using System.Data;

namespace DataSetDeletionExample
{
    public partial class Form1 : Form
    {
        Label label1;
        TextBox textbox1;
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(650, 300);
            this.Text = "ADO.NET NonQuery - Deletion";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* label1 */
            label1 = new Label();
            label1.AutoSize = true;
            label1.Text = "Existing Emp ID: ";
        }
    }
}
```

```
label1.Location = new Point(50, 50);
this.Controls.Add(label1);

/* textbox1 */
textbox1 = new TextBox();
textbox1.Size = new Size(200, 50);
textbox1.Location = new Point(300, 50);
this.Controls.Add(textbox1);

/* button1 */
button1 = new Button();
button1.AutoSize = true;
button1.Text = "Delete";
button1.Location = new Point(300, 120);
this.AcceptButton = button1;
button1.Click += Button1_Click;
this.Controls.Add(button1);

}

private void Button1_Click(object sender, EventArgs e)
{
    //create reference variables
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataAdapter adp;
    SqlCommandBuilder cmb;
    DataSet ds;
    DataTable dt;
    DataRow drow;

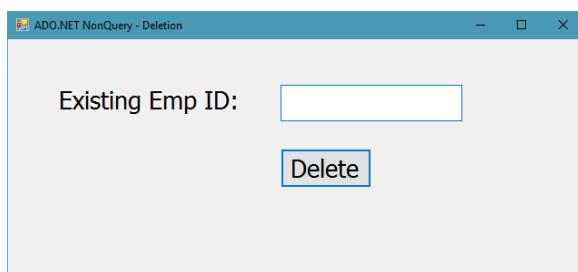
    //create objects
    cn = new SqlConnection();
    cmd = new SqlCommand();
    adp = new SqlDataAdapter();
    cmb = new SqlCommandBuilder();
    ds = new DataSet();
}
```

```
//call properties  
cn.ConnectionString = "data source=localhost; integrated  
security=yes; initial catalog=company";  
cmd.CommandText = "select * from Employees";  
cmd.Connection = cn;  
adp.SelectCommand = cmd;  
cmbDataAdapter = adp;  
  
//call methods  
adp.Fill(ds);  
dt = ds.Tables[0];  
  
//get datarow based on empid  
int n = Convert.ToInt32(textBox1.Text);  
string condition = "empid=" + n;  
drow = dt.Select(condition)[0];  
//delete the row  
drow.Delete();  
//save changes back to the database  
adp.Update(ds);  
MessageBox.Show("Deleted");  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter Existing EmplID and click on “Delete”.

Note: If any database connection problem, it shows exception (run time error).

N-Tier Architecture - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
```

```
go
```

```
use company
```

```
go
```

```
create table Employees(
```

```
EmpID int primary key,
```

```
EmpName nvarchar(max),
```

```
Salary decimal)
```

```
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
```

```
insert into Employees values(3, 'Jones', 6000)
```

```
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Solution

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.

- Select “Other Project Types” – “Visual Studio Solutions”.
- Select “Blank Solution”.
- Type the name as “CompanySolution”.
- Type the location as “C:\CSharp”.
- Click on OK.

Creating Data Access Layer

- Open Solution Explorer.
- Right click on the solution (CompanySolution) and click on “Add” – “New Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Class Library”.
- Type the name as “CompanyDataAccess”.
- Click on OK.

Creating Business Logic Layer

- Open Solution Explorer.
- Right click on the solution (CompanySolution) and click on “Add” – “New Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Class Library”.
- Type the name as “CompanyBusinessLogic”.
- Click on OK.

Creating Presentation Logic

- Open Solution Explorer.
- Right click on the solution (CompanySolution) and click on “Add” – “New Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#” – “Windows Forms Application”.
- Type the name as “CompanyPresentationLogic”.

- Click on OK.

Adding references:

- Open Solution Explorer.
- Right click on “CompanyBusinessLogic” project and click on “Add” – “Reference” – “Projects” – Check the checkbox “CompanyDataAccess” – OK.
- Right click on “CompanyPresentationLogic” project and click on “Add” – “Reference” – “Projects” – Check the checkbox “CompanyBusinessLogic” – OK.

Setting the Startup Project

- Open Solution Explorer.
- Right click on “CompanyPresentationLogic” project and click on “Set as Startup Project”.

CompanyDataAccess\Class1.cs

```
//Data Access Logic
using System;
using System.Data.SqlClient;

namespace CompanyDataAccess
{
    public class EmployeesDataAccess
    {
        public int InsertEmployee(int EmpID, string EmpName, decimal Salary)
        {
            SqlConnection cn;
            SqlCommand cmd;
            SqlParameter p1, p2, p3;

            cn = new SqlConnection();
            cmd = new SqlCommand();
            p1 = new SqlParameter();
            p2 = new SqlParameter();
            p3 = new SqlParameter();
        }
    }
}
```

```

        cn.ConnectionString = "data source=localhost; integrated
security=yes; initial catalog=company";
        cmd.CommandText = "insert into Employees values(@a, @b, @c)";
        cmd.Connection = cn;
        p1.ParameterName = "@a";
        p2.ParameterName = "@b";
        p3.ParameterName = "@c";
        p1.Value = EmpID;
        p2.Value = EmpName;
        p3.Value = Salary;
        cmd.Parameters.Add(p1);
        cmd.Parameters.Add(p2);
        cmd.Parameters.Add(p3);
        cn.Open();
        int n = cmd.ExecuteNonQuery();
        cn.Close();

        return n;
    }
}
}

```

CompanyBusinessLogic\Class1.cs

```

//Business Logic
using System;
using CompanyDataAccess;

namespace CompanyBusinessLogic
{
    public class EmployeesBusinessLogic
    {
        private int _empid;
        private string _empname;
        private decimal _salary;

        public int EmpID

```

```
{  
    set  
    {  
        if (value >= 0)  
        {  
            _empid = value;  
        }  
    }  
    get  
    {  
        return _empid;  
    }  
}  
  
public string EmpName  
{  
    set  
    {  
        if (value.Length <= 40)  
        {  
            _empname = value;  
        }  
    }  
    get  
    {  
        return _empname;  
    }  
}  
  
public decimal Salary  
{  
    set  
    {  
        if (value >= 1000 && value <= 5000)  
        {  
            _salary = value;  
        }  
    }  
}
```

```

        }
        get
        {
            return _salary;
        }
    }

    public string Insert()
    {
        EmployeesDataAccess da = new EmployeesDataAccess();
        int result = da.InsertEmployee(this.EmpID, this.EmpName,
this.Salary);
        if (result == 1)
        {
            return "Inserted";
        }
        else
        {
            return "Error in insertion";
        }
    }
}

```

CompanyPresentationLogic\Form1.cs

```

//Presentation Logic
using System;
using System.Drawing;
using System.Windows.Forms;
using CompanyBusinessLogic;

namespace CompanyPresentation
{
    public partial class Form1 : Form
    {
        Label lblHeading, lblEmployeeID, lblEmployeeName, lblSalary;
        TextBox txtEmployeeID, txtEmployeeName, txtSalary;

```

```
Button btnInsert;

public Form1()
{
    InitializeComponent();
    /* form properties */
    this.Text = "N-Tier Architecture";
    this.Font = new Font("Tahoma", 20);
    this.Size = new Size(600, 500);
    this.StartPosition = FormStartPosition.CenterScreen;
    /* lblHeading */
    lblHeading = new Label();
    lblHeading.Text = "New Employee";
    lblHeading.Font = new Font("Comic Sans MS", 30);
    lblHeading.AutoSize = true;
    lblHeading.BackColor = Color.Gold;
    lblHeading.Location = new Point(150, 50);
    this.Controls.Add(lblHeading);
    /* lblEmployeeID */
    lblEmployeeID = new Label();
    lblEmployeeID.Text = "Employee ID: ";
    lblEmployeeID.AutoSize = true;
    lblEmployeeID.Location = new Point(50, 150);
    this.Controls.Add(lblEmployeeID);
    /* lblEmployeeName */
    lblEmployeeName = new Label();
    lblEmployeeName.Text = "Employee Name: ";
    lblEmployeeName.AutoSize = true;
    lblEmployeeName.Location = new Point(50, 220);
    this.Controls.Add(lblEmployeeName);
    /* lblSalary */
    lblSalary = new Label();
    lblSalary.Text = "Salary: ";
    lblSalary.AutoSize = true;
    lblSalary.Location = new Point(50, 290);
    this.Controls.Add(lblSalary);
    /* txtEmployeeID */
}
```

```

txtEmployeeID = new TextBox();
txtEmployeeID.Size = new Size(200, 40);
txtEmployeeID.Location = new Point(300, 150);
this.Controls.Add(txtEmployeeID);
/* txtEmployeeName */
txtEmployeeName = new TextBox();
txtEmployeeName.Size = new Size(200, 40);
txtEmployeeName.Location = new Point(300, 220);
this.Controls.Add(txtEmployeeName);
/* txtSalary */
txtSalary = new TextBox();
txtSalary.Size = new Size(200, 40);
txtSalary.Location = new Point(300, 290);
this.Controls.Add(txtSalary);
/* btnInsert */
btnInsert = new Button();
btnInsert.AutoSize = true;
btnInsert.Location = new Point(300, 360);
btnInsert.Text = "Insert";
btnInsert.Click += BtnInsert_Click;
this.Controls.Add(btnInsert);
this.AcceptButton = btnInsert;
}

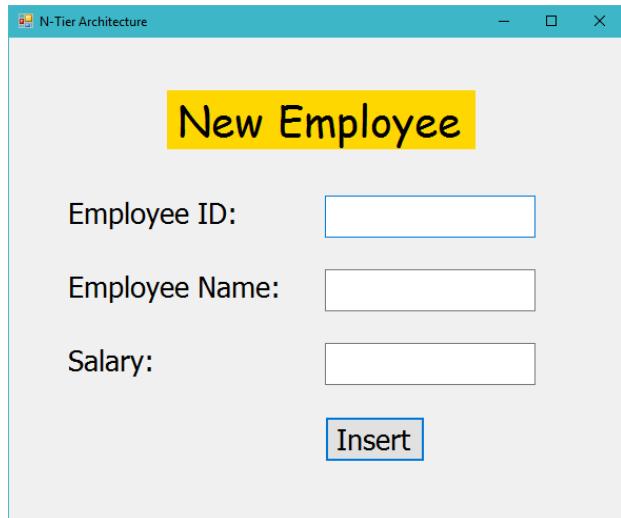
private void BtnInsert_Click(object sender, EventArgs e)
{
    EmployeesBusinessLogic b = new EmployeesBusinessLogic();
    b.EmpID = Convert.ToInt32(txtEmployeeID.Text);
    b.EmpName = txtEmployeeName.Text;
    b.Salary = Convert.ToDecimal(txtSalary.Text);
    string msg = b.Insert();
    MessageBox.Show(msg);
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

HARSHA

Output

Enter some new emp id, emp name and salary and click on “Insert”.

Note: If any database connection problem, it shows exception (run time error).

LINQ to SQL - Example

Creating Database

- **Note:** Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
```

Salary decimal)

go

insert into Employees values(1, 'Scott', 4000)

insert into Employees values(2, 'Allen', 5000)

insert into Employees values(3, 'Jones', 6000)

go

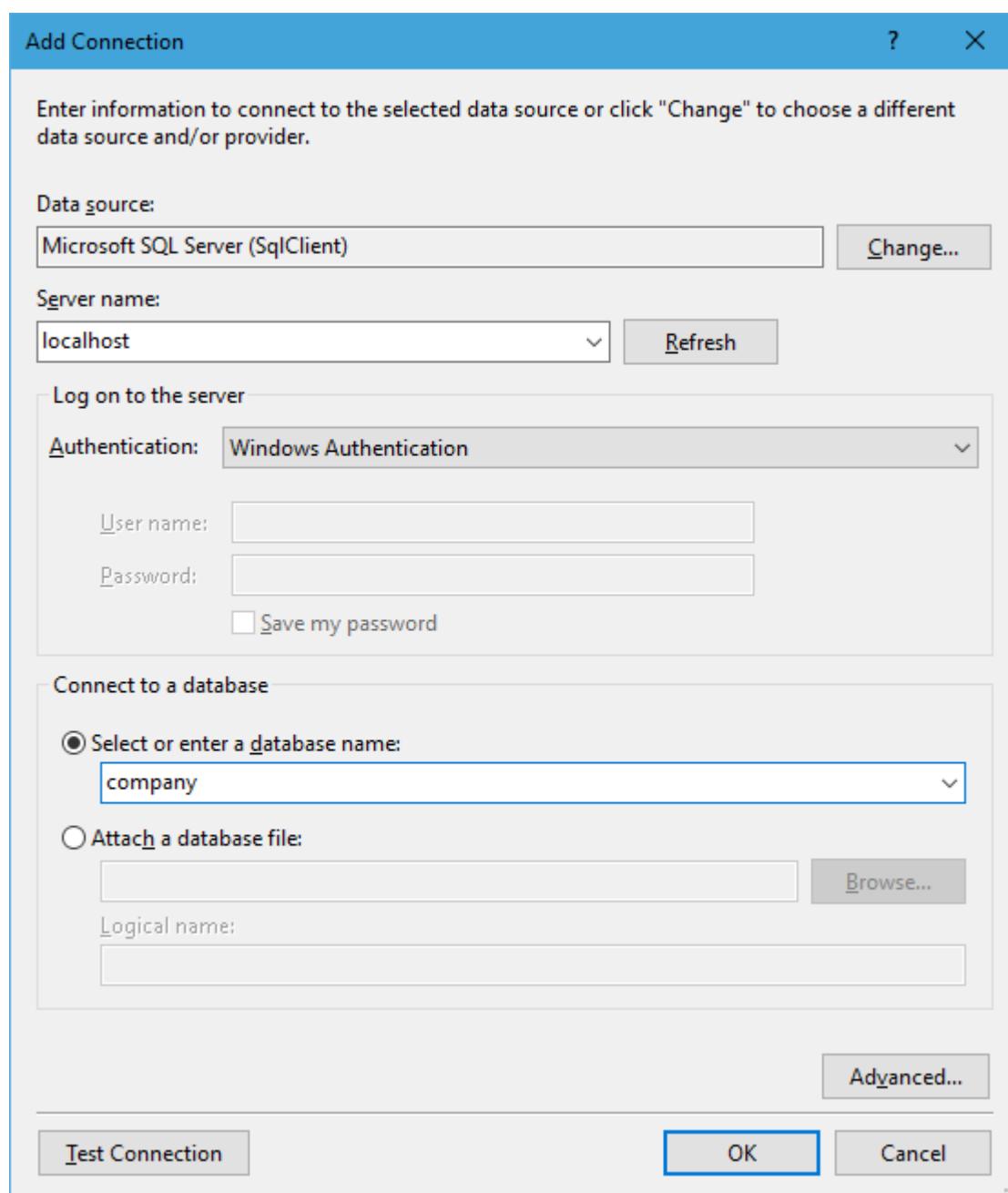
- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

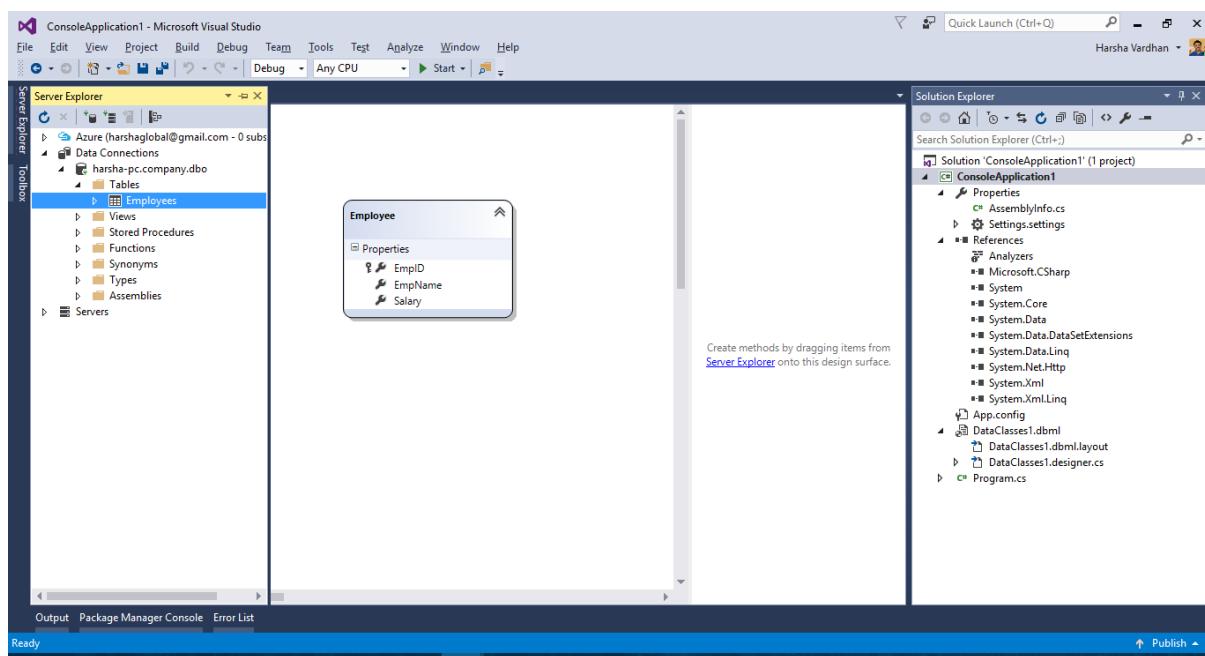
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “LinqToSqlExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “LinqToSqlExample”.
- Click on OK.

Creating “DBML” file:

- Open Solution Explorer.
- Right click on the project (LinqToSqlExample) and click on “Add” – “New Item” – “Data” – “LINQ to SQL Classes”.
- Type the name as “DataClasses1.dbml” (Database Markup Language).
- Click on “Add”.
- Go to “View” menu – “Server Explorer”.
- Right click on “Data Connections” and click on “Add Connection”.
- Type the server name as “localhost”.
- Select the authentication as “Windows Authentication”.
- Select the database name as “company”.
- Click on OK.



- Expand “company.dbo” – “Tables” – “Employees”.
- Drag and drop “Employees” table from Server Explorer into “Object Relational Designer” of “DataClasses1.dbml” file.
- It generate two classes called “DataClasses1DataContext” and “Employee”.



- Go to “Build” menu – “Build Solution”.

Form1.cs

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace LinqToSqlExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
        }
    }
}

```

```

this.Size = new Size(500, 300);
this.Text = "LINQ to SQL";
this.StartPosition = FormStartPosition.CenterScreen;

/* button1 */
button1 = new Button();
button1.Text = "Get Data";
button1.AutoSize = true;
button1.Location = new Point(170, 100);
button1.Click += button1_Click;
this.Controls.Add(button1);

}

private void button1_Click(object sender, EventArgs e)
{
    //create reference variables
    DataClasses1DataContext db;

    //create objects
    db = new DataClasses1DataContext();

    //LINQ query
    List<Employee> emps = db.Employees.ToList();

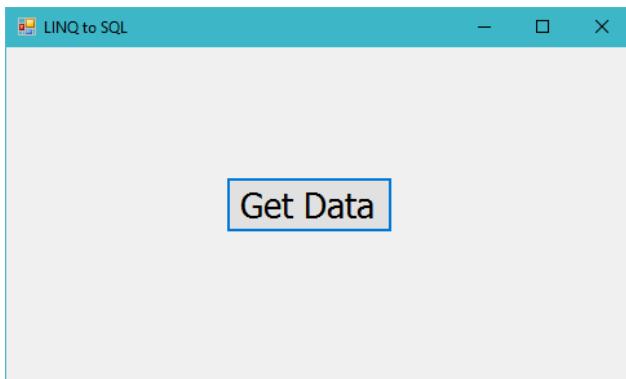
    //read data row-by-row
    for (int i = 0; i < emps.Count; i++)
    {
        Employee emp = emps[i];
        MessageBox.Show(emp.EmpID + ", " + emp.EmpName + ", " +
emp.Salary);
    }
}
}
}

```

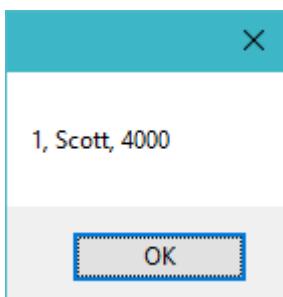
Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

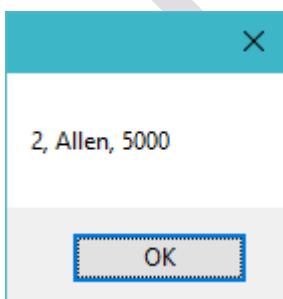
Output



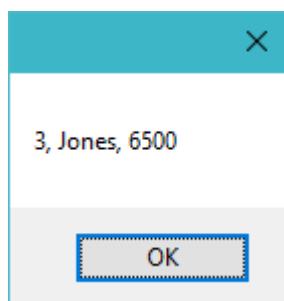
Click on "Get Data".



Click on OK.



Click on OK.



Note: If any database connection problem, it shows exception (run time error).

HARSHA

C#.NET – WinForms – Entity Framework

Entity Framework - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company  
go  
use company  
go
```

```
create table Employees(  
EmpID int primary key,  
EmpName nvarchar(max),  
Salary decimal)  
go
```

```
insert into Employees values(1, 'Scott', 4000)  
insert into Employees values(2, 'Allen', 5000)  
insert into Employees values(3, 'Jones', 6000)  
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

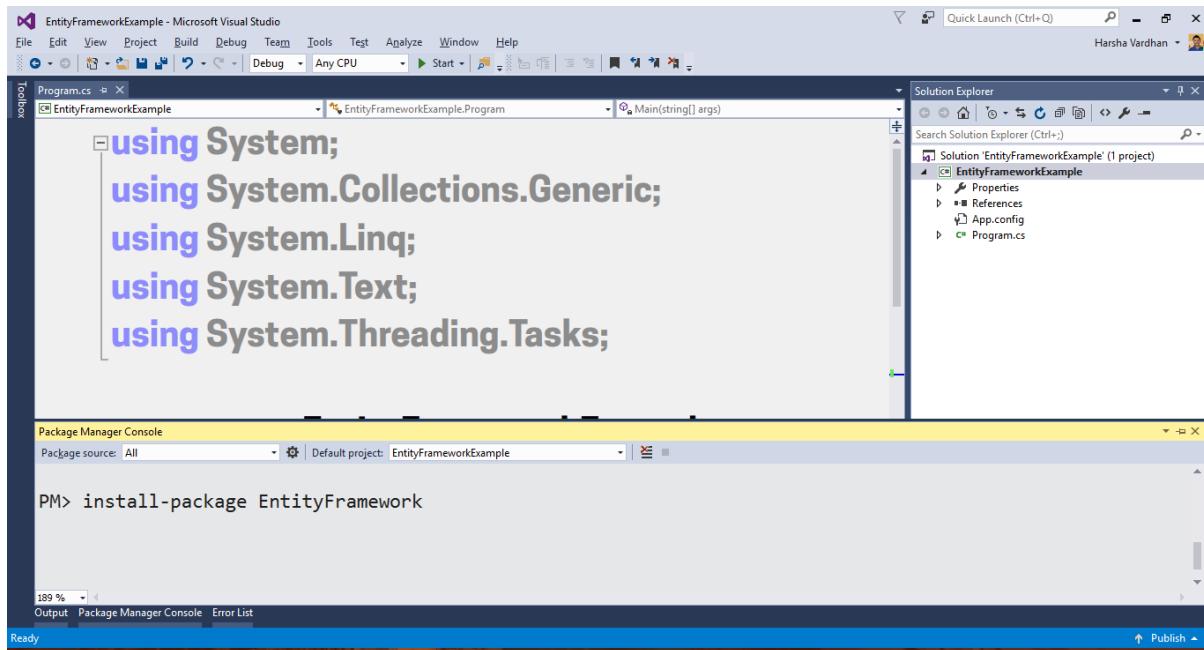
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “EntityFrameworkExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EntityFrameworkExample”.
- Open Solution Explorer. Right click on the project (EntityFrameworkExample) and click on “Add” – “New Item” – “Code” – “Class”. Type the filename as “Employee.cs”. Click on “Add”.
- Click on OK. Right click on Form1 and click on “View Code”.

Harsh

Installing "EntityFramework" NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

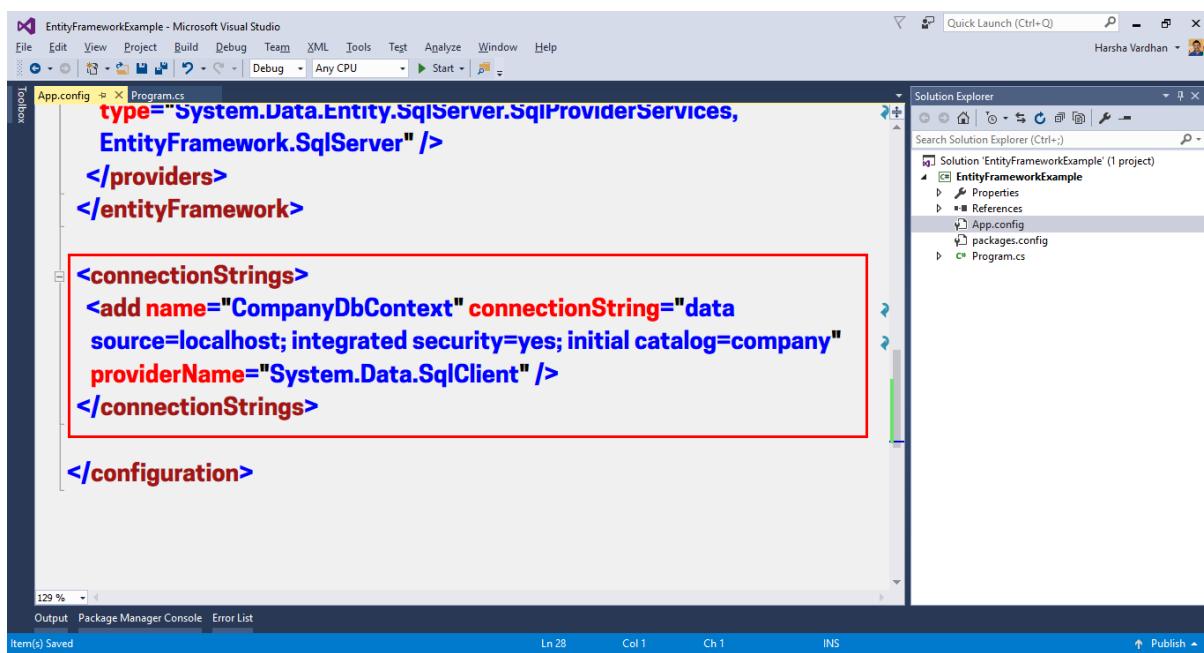
Install-package EntityFramework



Adding "ConnectionString" in "App.Config":

- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

```
<connectionStrings>
  <add name="CompanyDbContext" connectionString="data
    source=localhost; integrated security=yes; initial
    catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>
```



Employee.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.Data.Entity;

namespace EntityFrameworkExample
{
    public class Employee
    {
        [Key]
        public int EmpID { get; set; }
        public string EmpName { get; set; }
        public decimal Salary { get; set; }
    }

    public class CompanyDbContext : DbContext
    {
        public DbSet<Employee> Employees { get; set; }
    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace EntityFrameworkExample
{
    public partial class Form1 : Form
    {
        Button button1;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(500, 300);
            this.Text = "ADO.NET Entity Framework";
            this.StartPosition = FormStartPosition.CenterScreen;

            /* button1 */
            button1 = new Button();
            button1.Text = "Get Data";
            button1.AutoSize = true;
            button1.Location = new Point(170, 100);
            button1.Click += button1_Click;
            this.Controls.Add(button1);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //create reference variables
```

```

CompanyDbContext db;

//create objects
db = new CompanyDbContext();

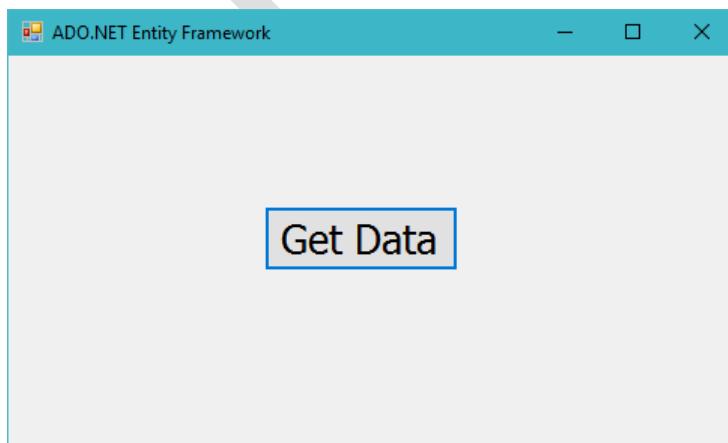
//LINQ query
//List<Employee> emps = db.Employees.ToList();
//List<Employee> emps = db.Employees.Where(temp => temp.Salary
>= 2000).ToList();
List<Employee> emps = db.Employees.OrderBy(temp =>
temp.Salary).ToList();

//read data row-by-row
for (int i = 0; i < emps.Count; i++)
{
    Employee emp = emps[i];
    MessageBox.Show(emp.EmpID + ", " + emp.EmpName + ", " +
emp.Salary);
}
}
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output

Click on “Get Data”. It shows all records one-by-one.

Note: If any database connection problem, it shows exception (run time error).

Entity Framework – Insertion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
create table Employees(
EmpID int primary key,
EmpName nvarchar(max),
Salary decimal)
go
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

Creating Project

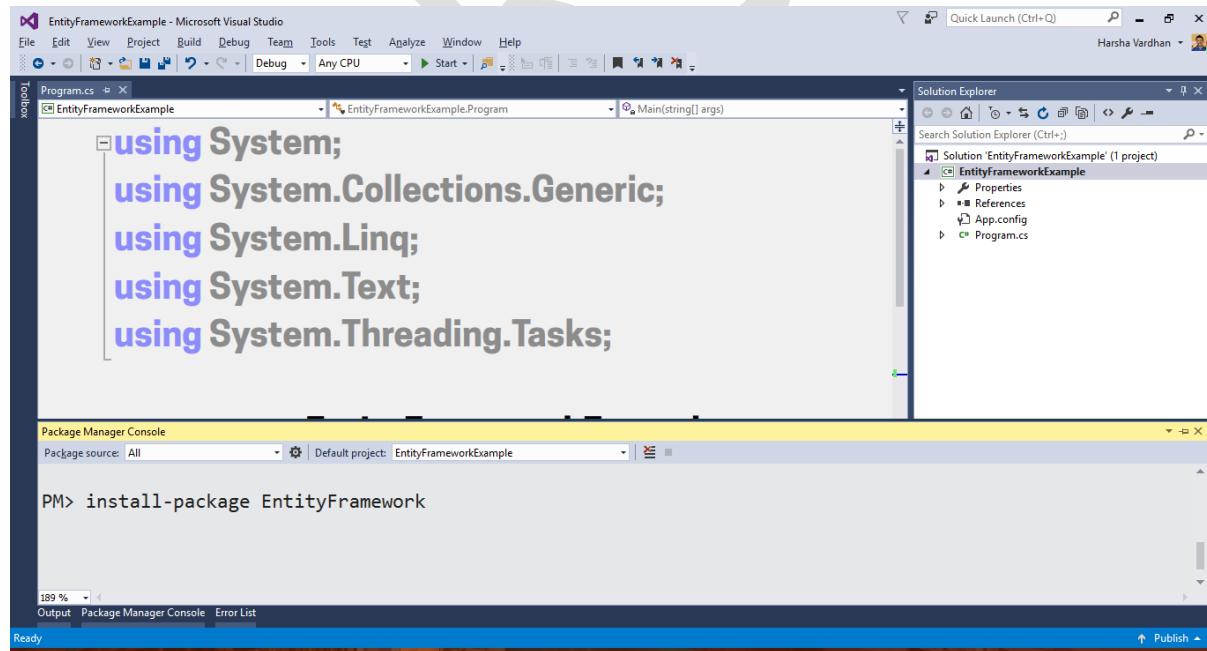
- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.

- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “EfInsertionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EfInsertionExample”.
- Open Solution Explorer. Right click on the project (EfInsertionExample) and click on “Add” – “New Item” – “Code” – “Class”. Type the filename as “Employee.cs”. Click on “Add”.
- Click on OK. Right click on Form1 and click on “View Code”.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

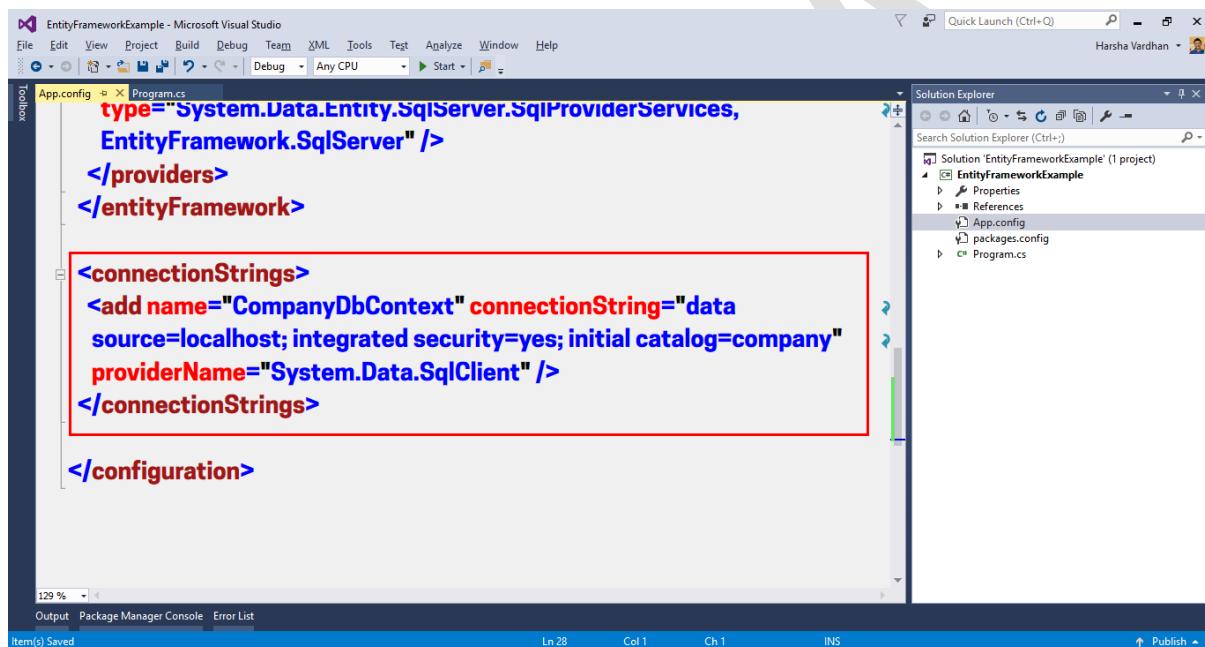
Install-package EntityFramework



Adding "ConnectionString" in "App.Config":

- Go to "App.Config" (If exists already).
- Scroll down to end of "App.Config" file and add "ConnectionStrings" as follows:

```
<connectionStrings>
  <add name="CompanyDbContext" connectionString="data
  source=localhost; integrated security=yes; initial
  catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>
```

**Employee.cs**

```

using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;

namespace EfInsertionExample
{
  public class Employee
  {
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
  }
}
```

```

public int EmpID { get; set; }
public string EmpName { get; set; }
public decimal Salary { get; set; }
}

public class CompanyDbContext : DbContext
{
    public DbSet<Employee> Employees { get; set; }
}
}

```

Form1.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Drawing;
using System.Windows.Forms;

namespace EfInsertionExample
{
    public partial class Form1 : Form
    {
        Label lblEmpID, lblEmpName, lblSalary;
        TextBox txtEmpID, txtEmpName, txtSalary;
        Button btnOK;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 20);
            this.Size = new Size(600, 400);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Text = "ADO.NET Entity Framework - Insertion";

            /* lblEmpID */
        }
    }
}

```

```
lblEmpID = new Label();
lblEmpID.Text = "Emp ID: ";
lblEmpID.AutoSize = true;
lblEmpID.Location = new Point(50, 50);
this.Controls.Add(lblEmpID);

/* lblEmpName */
lblEmpName = new Label();
lblEmpName.Text = "Emp Name: ";
lblEmpName.AutoSize = true;
lblEmpName.Location = new Point(50, 120);
this.Controls.Add(lblEmpName);

/* lblSalary */
lblSalary = new Label();
lblSalary.Text = "Salary: ";
lblSalary.AutoSize = true;
lblSalary.Location = new Point(50, 190);
this.Controls.Add(lblSalary);

/* txtEmpID */
txtEmpID = new TextBox();
txtEmpID.Size = new Size(200, 50);
txtEmpID.Location = new Point(270, 50);
this.Controls.Add(txtEmpID);

/* txtEmpName */
txtEmpName = new TextBox();
txtEmpName.Size = new Size(200, 50);
txtEmpName.Location = new Point(270, 120);
this.Controls.Add(txtEmpName);

/* txtSalary */
txtSalary = new TextBox();
txtSalary.Size = new Size(200, 50);
txtSalary.Location = new Point(270, 190);
this.Controls.Add(txtSalary);
```

```
/* btnOK */
btnOK = new Button();
btnOK.Size = new Size(200, 50);
btnOK.Text = "Insert";
btnOK.Location = new Point(270, 270);
this.AcceptButton = btnOK;
btnOK.Click += BtnOK_Click;
this.Controls.Add(btnOK);
}

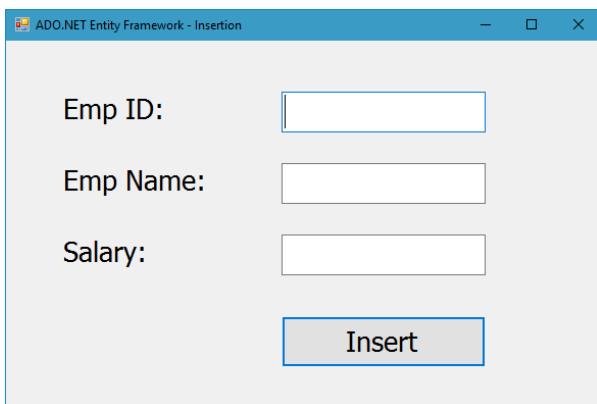
private void BtnOK_Click(object sender, EventArgs e)
{
    //insertion
    CompanyDbContext db = new CompanyDbContext();
    Employee emp = new Employee();
    emp.EmpID = Convert.ToInt32(txtEmpID.Text);
    emp.EmpName = txtEmpName.Text;
    emp.Salary = Convert.ToDecimal(txtSalary.Text);
    db.Employees.Add(emp);
    db.SaveChanges();

    MessageBox.Show("Inserted");
}
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some emp id, emp name and salary and click on “Insert”.

Note: If any database connection problem, it shows exception (run time error).

Entity Framework – Updation - Example

Creating Database

- **Note:** Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

```
create database company
go
use company
go
```

```
create table Employees(
    EmpID int primary key,
    EmpName nvarchar(max),
    Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
```

```
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

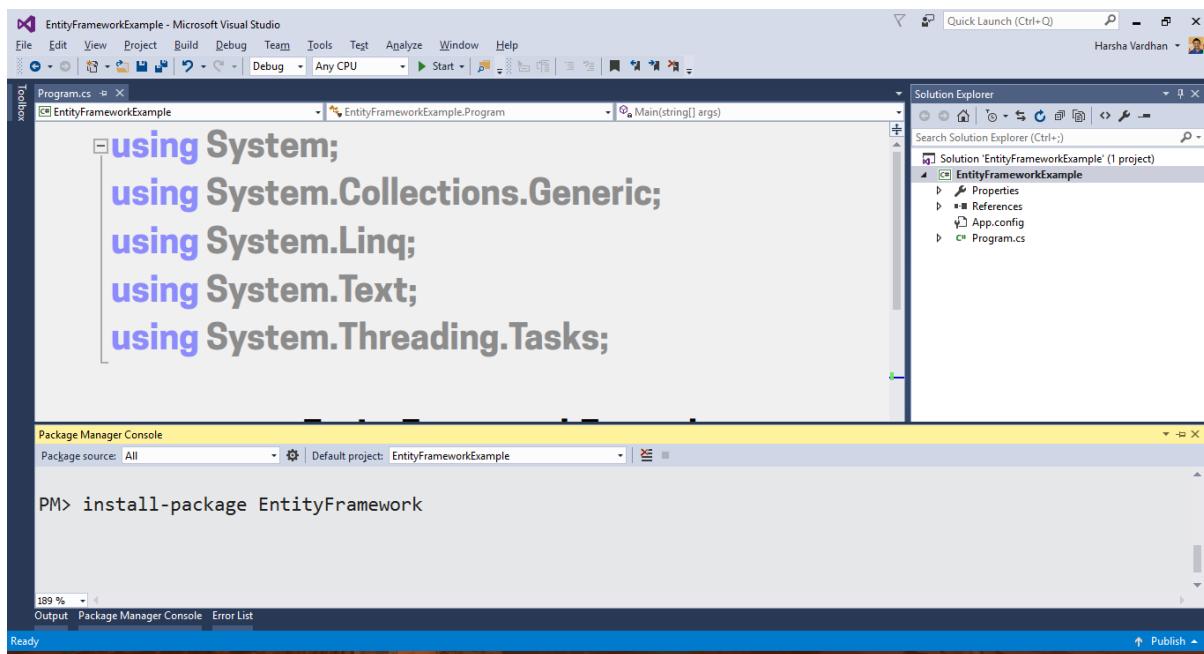
Creating Project

- Open Visual Studio 2019. Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”. Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “EfUpdationExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EfUpdationExample”.
- Open Solution Explorer. Right click on the project (EfUpdationExample) and click on “Add” – “New Item” – “Code” – “Class”. Type the filename as “Employee.cs”. Click on “Add”.
- Click on OK. Right click on Form1 and click on “View Code”.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

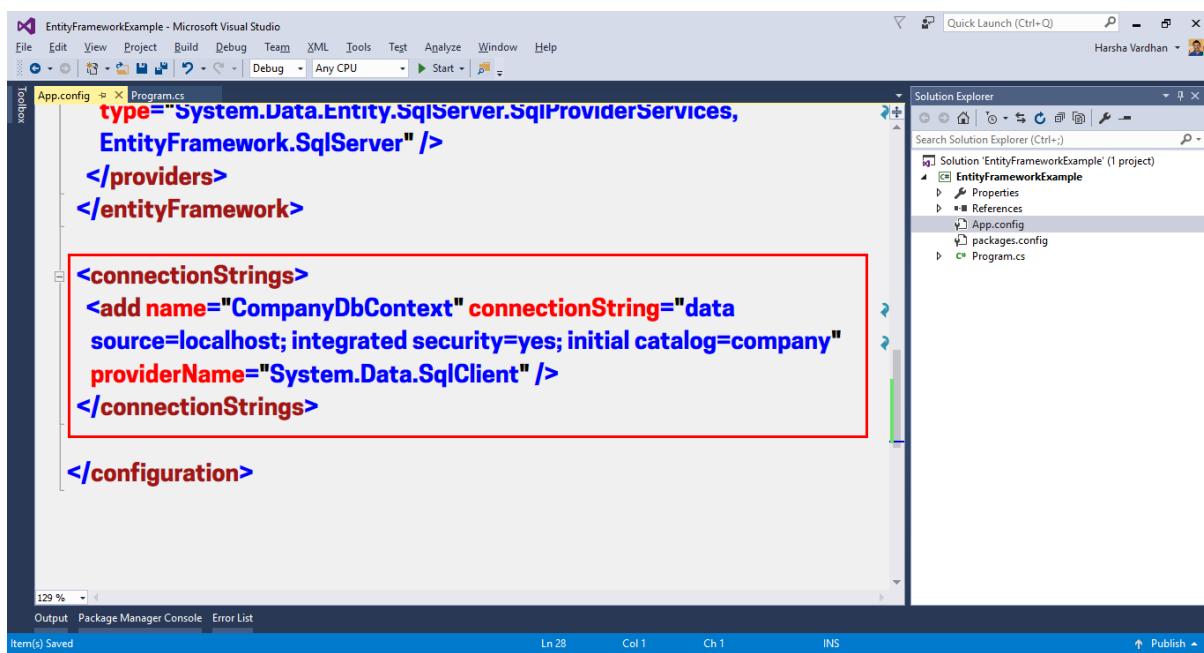
```
Install-package EntityFramework
```



Adding "ConnectionString" in "App.Config":

- Go to "App.Config" (If exists already).
- Scroll down to end of "App.Config" file and add "ConnectionStrings" as follows:

```
<connectionStrings>
  <add name="CompanyDbContext" connectionString="data
  source=localhost; integrated security=yes; initial
  catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>
```

**Employee.cs**

```

using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;

namespace EfUpdationExample
{
  public class Employee
  {
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.None)]
    public int EmpID { get; set; }
    public string EmpName { get; set; }
    public decimal Salary { get; set; }
  }

  public class CompanyDbContext : DbContext
  {
    public DbSet<Employee> Employees { get; set; }
  }
}

```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Drawing;
using System.Windows.Forms;

namespace EfUpdationExample
{
    public partial class Form1 : Form
    {
        Label lblEmpID, lblEmpName, lblSalary;
        TextBox txtEmpID, txtEmpName, txtSalary;
        Button btnOK;

        public Form1()
        {
            InitializeComponent();

            /* form properties */
            this.Font = new Font("Tahoma", 17);
            this.Size = new Size(600, 400);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.Text = "ADO.NET Entity Framework - Updation";
            /* lblEmpID */
            lblEmpID = new Label();
            lblEmpID.Text = "Existing Emp ID: ";
            lblEmpID.AutoSize = true;
            lblEmpID.Location = new Point(50, 50);
            this.Controls.Add(lblEmpID);
            /* lblEmpName */
            lblEmpName = new Label();
            lblEmpName.Text = "Emp Name: ";
            lblEmpName.AutoSize = true;
            lblEmpName.Location = new Point(50, 120);
            this.Controls.Add(lblEmpName);
```

```

/* lblSalary */
lblSalary = new Label();
lblSalary.Text = "Salary: ";
lblSalary.AutoSize = true;
lblSalary.Location = new Point(50, 190);
this.Controls.Add(lblSalary);

/* txtEmpID */
txtEmpID = new TextBox();
txtEmpID.Size = new Size(200, 50);
txtEmpID.Location = new Point(270, 50);
this.Controls.Add(txtEmpID);

/* txtEmpName */
txtEmpName = new TextBox();
txtEmpName.Size = new Size(200, 50);
txtEmpName.Location = new Point(270, 120);
this.Controls.Add(txtEmpName);

/* txtSalary */
txtSalary = new TextBox();
txtSalary.Size = new Size(200, 50);
txtSalary.Location = new Point(270, 190);
this.Controls.Add(txtSalary);

/* btnOK */
btnOK = new Button();
btnOK.Size = new Size(200, 50);
btnOK.Text = "Update";
btnOK.Location = new Point(270, 270);
this.AcceptButton = btnOK;
btnOK.Click += BtnOK_Click;
this.Controls.Add(btnOK);

}

private void BtnOK_Click(object sender, EventArgs e)
{
    //updation
    CompanyDbContext db = new CompanyDbContext();
    int n = Convert.ToInt32(txtEmpID.Text);
    Employee emp = db.Employees.Where(temp => temp.EmpID ==
n).FirstOrDefault();
}

```

```

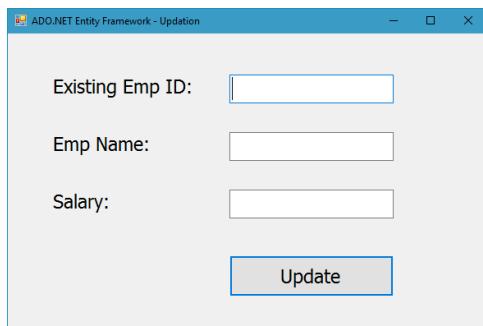
        emp.EmpName = txtEmpName.Text;
        emp.Salary = Convert.ToDecimal(txtSalary.Text);
        db.SaveChanges();
        MessageBox.Show("Updated");
    }
}
}
}

```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some existing emp id, new emp name and new salary and click on “Update”.

Note: If any database connection problem, it shows exception (run time error).

Entity Framework – Deletion - Example

Creating Database

- Note: Ignore this step, if you have created “company” database already.
- Open SQL Server Management Studio. Click on “Connect”.
- Click on “New Query”.
- Type the following code:

create database company

```
go
use company
go
```

```
create table Employees(
    EmpID int primary key,
    EmpName nvarchar(max),
    Salary decimal)
go
```

```
insert into Employees values(1, 'Scott', 4000)
insert into Employees values(2, 'Allen', 5000)
insert into Employees values(3, 'Jones', 6000)
go
```

- Click on “Execute” button. It shows “Query executed successfully” in the status bar.

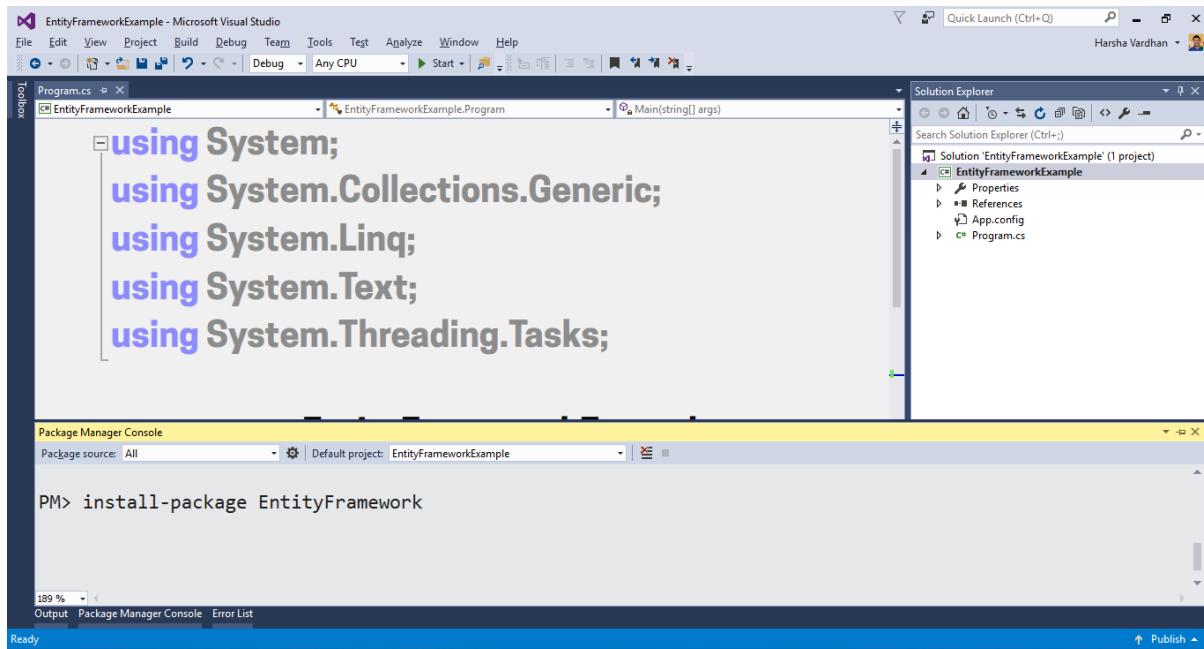
Creating Project

- Open Visual Studio 2019.
- Go to “File” – “New” – “Project”.
- Select “.NET Framework 4.8”.
- Select “Visual C#”.
- Select “Windows Forms Application”.
- Type the project name as “EfDeletionExample”.
- Type the location as “C:\CSharp”.
- Type the solution name as “EfDeletionExample”.
- Open Solution Explorer. Right click on the project (EfDeletionExample) and click on “Add” – “New Item” – “Code” – “Class”. Type the filename as “Employee.cs”. Click on “Add”.
- Click on OK. Right click on Form1 and click on “View Code”.

Installing “EntityFramework” NuGet Package

- Go to “Tools” menu – “NuGet Package Manager” - “Package Manager Console”.
- Type the command and press Enter.

Install-package EntityFramework



Adding “ConnectionString” in “App.Config”:

- Go to “App.Config” (If exists already).
- Scroll down to end of “App.Config” file and add “ConnectionStrings” as follows:

```
<connectionStrings>
  <add name="CompanyDbContext" connectionString="data
  source=localhost; integrated security=yes; initial
  catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>
```

C#.NET 8.0



A screenshot of Microsoft Visual Studio 2022 showing the configuration file `App.config`. The code defines the Entity Framework provider and a connection string named `CompanyDbContext` for a local SQL Server database.

```
<entityFramework>
  <providers>
    <provider name="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
  </providers>
</entityFramework>

<connectionStrings>
  <add name="CompanyDbContext" connectionString="data source=localhost; integrated security=yes; initial catalog=company" providerName="System.Data.SqlClient" />
</connectionStrings>

</configuration>
```

The `App.config` file is selected in the Solution Explorer. The code editor shows syntax highlighting for XML tags and C# attributes.

Employee.cs

```

using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;

namespace EfDeletionExample
{
    public class Employee
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public int EmpID { get; set; }
        public string EmpName { get; set; }
        public decimal Salary { get; set; }
    }
    public class CompanyDbContext : DbContext
    {
        public DbSet<Employee> Employees { get; set; }
    }
}

```

Form1.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Drawing;
using System.Windows.Forms;

namespace EfDeletionExample
{
    public partial class Form1 : Form
    {
        Label lblEmpID;
        TextBox txtEmpID;
        Button btnOK;
        public Form1()
    }
}

```

```

{
    InitializeComponent();

    /* form properties */
    this.Font = new Font("Tahoma", 17);
    this.Size = new Size(600, 250);
    this.StartPosition = FormStartPosition.CenterScreen;
    this.Text = "ADO.NET Entity Framework - Deletion";
    /* lblEmpID */
    lblEmpID = new Label();
    lblEmpID.Text = "Existing Emp ID: ";
    lblEmpID.AutoSize = true;
    lblEmpID.Location = new Point(50, 50);
    this.Controls.Add(lblEmpID);
    /* txtEmpID */
    txtEmpID = new TextBox();
    txtEmpID.Size = new Size(200, 50);
    txtEmpID.Location = new Point(270, 50);
    this.Controls.Add(txtEmpID);
    /* btnOK */
    btnOK = new Button();
    btnOK.Size = new Size(200, 50);
    btnOK.Text = "Delete";
    btnOK.Location = new Point(270, 100);
    this.AcceptButton = btnOK;
    btnOK.Click += BtnOK_Click;
    this.Controls.Add(btnOK);
}
private void BtnOK_Click(object sender, EventArgs e)
{
    //updation
    CompanyDbContext db = new CompanyDbContext();
    int n = Convert.ToInt32(txtEmpID.Text);
    Employee emp = db.Employees.Where(temp => temp.EmpID ==
n).FirstOrDefault();
    db.Employees.Remove(emp);
    db.SaveChanges();
}

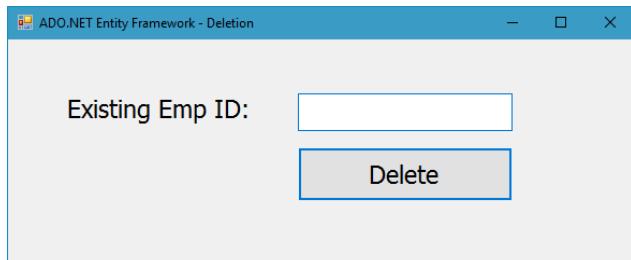
```

```
    MessageBox.Show("Deleted");  
}  
}  
}
```

Running the Project

- Go to “Debug” menu and click on “Start Debugging”.

Output



Enter some existing emp id and click on “Delete”.

Note: If any database connection problem, it shows exception (run time error).