Antony Roshan

**CSHarp - 3**

**Abstract Class**

using System;

namespace ConsoleApp1

{

abstract class Person //abstract class

{

public string Name { get; set; }

public int Age { get; set; }

public abstract void work(); //abstract method

}

class Student : Person

{

public override void work()

{

Console.WriteLine("It studies....");

}

}

class Employee : Person

{

public double Salary { get; set; }

public string Designation { get; set; }

public override void work()

{

Console.WriteLine("It earns....");

}

}

class Manager : Employee

{

public sealed override void work()

{

Console.WriteLine("It manages a team....");

}

}

class BranchManager : Manager

{

}

class program

{

static void Main(string[] args)

{

Person p1 = new Student();

p1.work();

p1 = new Employee();

p1.work();

p1 = new BranchManager();

p1.work();

Console.ReadLine();

}

}

}

**Interface**

* Gives complete abstraction
* Provide multiple inheritance

using System;

namespace ConsoleApp1

{

interface IInterface1

{

void Method1();

void Method2();

}

interface IInterface2

{

void Method2();

}

class MyClass1

{

public void M1()

{

Console.WriteLine("M1 from MyClass2");

}

}

class MyClass2 : MyClass1,IInterface1,IInterface2

{

public void Method1()

{

Console.WriteLine("Method 1");

}

void IInterface1.Method2()

{

Console.WriteLine("Method 2 from IInterace1");

}

void IInterface2.Method2()

{

Console.WriteLine("Method 2 from IInterace2");

}

}

class program

{

static void Main(string[] args)

{

MyClass2 m2 = new MyClass2();

IInterface1 i1 = new MyClass2();

i1.Method2();

IInterface1 i2 = new MyClass2();

i2.Method2();

Console.ReadLine();

}

}

}

**Sealed Class**

using System;

namespace ConsoleApp1

{

sealed class MyClass1 //class is sealed

{

public void SampleMethod()

{

Console.WriteLine("Sample Method in MyClass1");

}

}

class MyClass2 : MyClass1 //base class is sealed so canot inherit

{

}

class program

{

static void Main(string[] args)

{

Console.ReadLine();

}

}

}

**Generic Class**

using System;

namespace ConsoleApp1

{

class MyClass<T1,T2>

{

public T1 MyProperty { get; set; }

public void Method(T1 p1,T2 p2)

{

Console.WriteLine(p1);

Console.WriteLine(p2);

}

}

class program

{

static void Main(string[] args)

{

MyClass<int, string> obj1= new MyClass<int, string>();

obj1.MyProperty = 10;

obj1.Method(10, "CSharp");

Console.ReadLine();

}

}

}

**Structures**

using System;

namespace ConsoleApp1

{

struct MyStruct

{

public int Val { get; set; }

}

class program

{

static void Main(string[] args)

{

MyStruct m1 = new MyStruct();

m1.Val = 10;

MyStruct m2 = m1;

Console.WriteLine(m2.Val);

m1.Val = 20;

Console.WriteLine(m2.Val);

Console.WriteLine(m1.Val);

Console.ReadLine();

}

}

}

**Enum**

using System;

namespace ConsoleApp1

{

enum Gender

{

Female =1,

Male = 2,

Unknown = 3

}

class Person

{

public string Name { get; set; }

public int Age { get; set; }

public Gender Gender { get; set; }

}

class program

{

static void Main(string[] args)

{

Person p1=new Person { Name = "Anadi", Age = 28, Gender = Gender.Male };

int gen = (int)p1.Gender;

Console.WriteLine(gen);

Console.ReadLine();

}

}

}

**Delegates**

* Is a reference type variable that holds the reference to a method.
* Used for implementing events and the call-back methods.
* Implicitly derived from the System.Delegate class
* Delegate holding the reference of multiple methods is called multicast delegate

using System;

namespace ConsoleApp1

{

delegate int MyDel(int x, int y);

class Person

{

public static int Add(int x, int y)

{

return x + y;

}

public static int Multiply(int x, int y)

{

return x \* y;

}

static void Main(string[] args)

{

MyDel del = new MyDel(Add);

int result=del(10, 30);

Console.WriteLine(result);

del += Multiply;

result =del(10, 30);

Console.WriteLine(result);

Console.ReadLine();

}

}

}

**Generic Delegate**

using System;

namespace ConsoleApp1

{

delegate int MyDel(int x, int y);

class Person

{

public static void SampleMethod(string name)

{

Console.WriteLine("Welcome "+name);

}

public static int Add(int x, int y)

{

return x + y;

}

public static bool Login(string uid)

{

if (uid == "anadi")

return true;

else

return false;

}

static void Main(string[] args)

{

Action<string> action1 = SampleMethod;

Func<int, int, int> func1 = Add;

Predicate<string> pred1 = Login;

action1("CSharp");

Console.ReadLine();

}

}

}

**Anonymous Method**

* Anonymous method are the methods without a name
* Method can be invoked directly by delegate

using System;

namespace ConsoleApp1

{

delegate int Del();

class Program

{

static void Main(string[] args)

{

Del del = delegate()

{

Console.WriteLine("This is anonymous method");

Console.ReadLine();

};

}

}

}

**Lambda Expressions**

using System;

namespace ConsoleApp1

{

delegate int Del(int x, int y);

class Program

{

static void Main(string[] args)

{

Del del = (x, y) => x \* y;

int result = del(4, 6);

Console.WriteLine(result);

Console.ReadLine();

}

}

}

**Introduction to GUI**

**Double Click**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

MessageBox.Show("Welcome to windows Application");

}

private void Form1\_Load(object sender, EventArgs e)

{

button1.Text = "click me....";

}

}

}

**Close Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Form1\_Load(object sender, EventArgs e)

{

button1.Text = "Close Form";

}

}

}

**Dialog - Close Form Confirmation**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult dr = MessageBox.Show("Are you sure to close this form", "Confirm",

MessageBoxButtons.YesNo,MessageBoxIcon.Question);

if(dr == DialogResult.Yes)

this.Close();

}

private void Form1\_Load(object sender, EventArgs e)

{

button1.Text = "Close Form";

}

}

}

**Exception Handling**

* An error that arises during the execution of a program
* Program terminates abnormally as soon as it arises
* Programmatically exceptions can be handled
* System.Exception is base exception class in .NET
* try... catch... finally... throws... are the keywords associated with exception handling

**Divide by Zero**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

int num1 = int.Parse(textBox1.Text);

int num2 = int.Parse(textBox2.Text);

/\*float num1 = float.Parse(textBox1.Text);

float num2 = float.Parse(textBox2.Text);

if (num2 == 0)

throw new DivideByZeroException();\*/

int result = num1 / num2;

label3.Text = result.ToString();

}

catch(DivideByZeroException)

{

label3.Text = " Cannot divide by zero ";

}

catch(FormatException)

{

label3.Text = "Please enter only non decimal numbers";

}

catch (Exception ex)

{

label3.Text = ex.GetType() + ":" + ex.Message;

}

finally

{

}

}

}

}

**Custom Exception**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

float num1 = float.Parse(textBox1.Text);

float num2 = float.Parse(textBox2.Text);

if (num1 == 0)

throw new MyCustomException("Number cannot be zero");

if (num2 == 0)

throw new DivideByZeroException();

float result = num1 / num2;

label3.Text = result.ToString();

}

catch(DivideByZeroException)

{

label3.Text = " Cannot divide by zero ";

}

catch(FormatException)

{

label3.Text = "Please enter only non decimal numbers";

}

catch (Exception ex)

{

label3.Text = ex.GetType() + ":" + ex.Message;

}

finally

{

}

}

}

}

**Regular Expression**

* Is a pattern that could be matched against an input text
* .Net provides a regular expression engine that allows matching.
* Pattern consists of one or more-character literals, operators, or constructs.
* System.Text.RegularExpressions provides Regex class which handles regular express tasks

using System;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Text : Form

{

public Text()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Regex reg = new Regex(txt\_Pattern.Text);

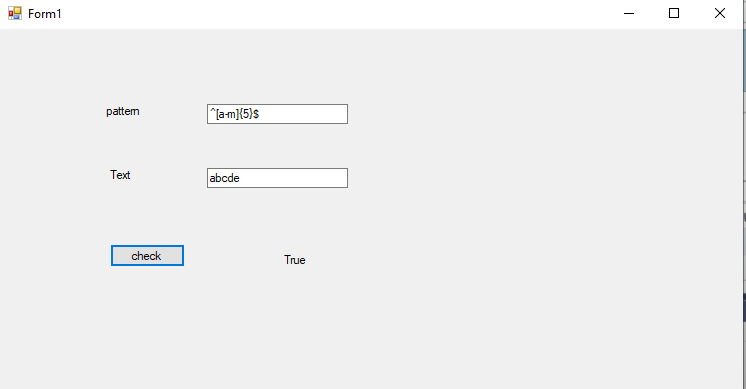
bool result = reg.IsMatch(txt\_Text.Text);

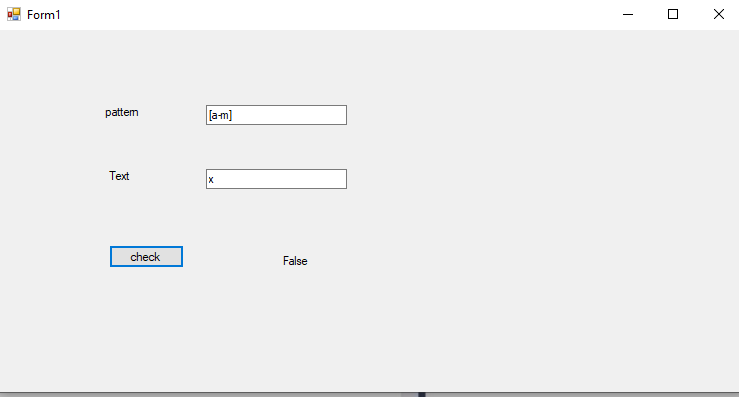
lbl\_result.Text = result.ToString();

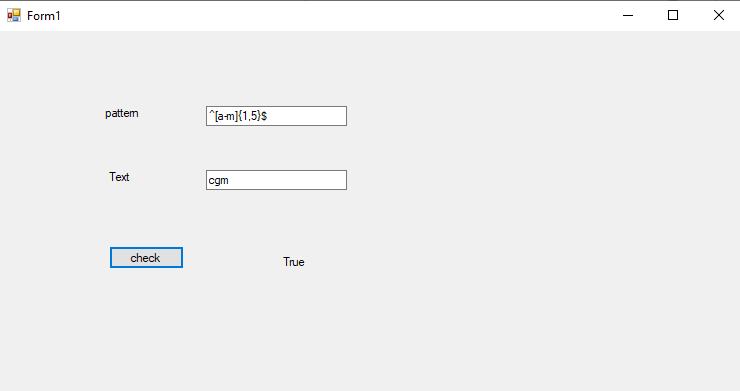
}

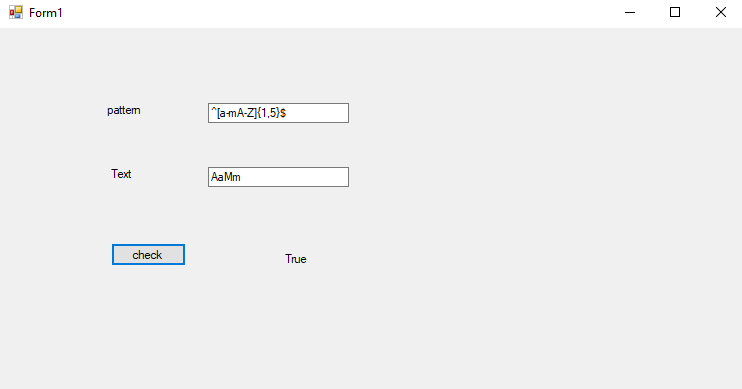
}

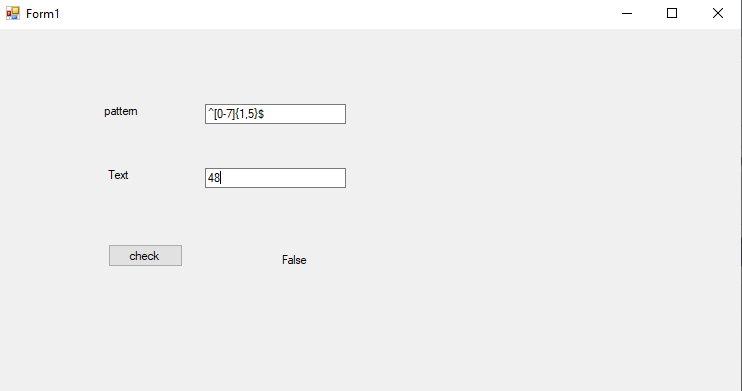
}

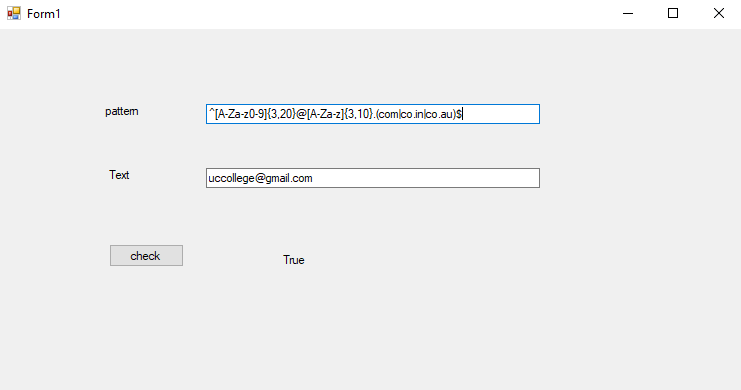












**Collections**

* Collection classes are specialized classes for data storage and retrieval
* Allocates memory dynamically
* All classes takes the data of object type
* System.Collections namespace provides various classes providing different data structure
* **Array List**
* **Stack**
* **Queue**
* **Hashtable**
* **SortedList**

**Array List**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form2 : Form

{

ArrayList List;

public Form2()

{

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

//List = new ArrayList(3);

List = new ArrayList(3);

List.Add("One");

List.Add("Two");

List.Add("Three");

List.Add("Four");

List.Add("Five");

int count = List.Count;

int capacity = List.Capacity;

List.TrimToSize();

capacity = List.Capacity;

MessageBox.Show("Count : " + count);

MessageBox.Show("Capacity : " + capacity);

//List.Remove("Three");

//List.RemoveAt(3);

//List.RemoveRange(1, 3);

//List.Sort();

//List.Reverse();

foreach (object item in List)

{

listBox1.Items.Add(item);

}

}

}

}

**Stack**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form3 : Form

{

Stack stack;

public Form3()

{

InitializeComponent();

}

private void Form3\_Load(object sender, EventArgs e)

{

stack = new Stack();

stack.Push("One");

stack.Push("Two");

stack.Push("Three");

stack.Push("Four");

stack.Push("Five");

stack.Pop();

object Top = stack.Peek();

MessageBox.Show(Top.ToString());

foreach (object item in stack)

{

listBox1.Items.Add(item);

}

}

}

}

**Queue**

using System;

using System.Collections;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form4 : Form

{

Queue queue;

public Form4()

{

InitializeComponent();

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void Form4\_Load(object sender, EventArgs e)

{

queue = new Queue();

queue.Enqueue("One");

queue.Enqueue("Two");

queue.Enqueue("Three");

queue.Enqueue("Four");

queue.Enqueue("Five");

queue.Dequeue();

foreach (var item in queue)

{

listBox1.Items.Add(item);

}

}

}

}

**Hashtable**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form5 : Form

{

Hashtable ht;

public Form5()

{

InitializeComponent();

}

private void Form5\_Load(object sender, EventArgs e)

{

ht = new Hashtable();

ht.Add("key1", "value 1");

ht.Add("abcd", "1234");

ht.Add("xyz", "1234");

ht.Add("ijkl", "Csharp");

ht.Add("2key", "point");

ht.Contains("abcd");

ht.Remove("key1");

foreach (DictionaryEntry item in ht)

{

listBox1.Items.Add(item.Key + " : " + item.Value);

}

}

}

}

**Generic Collections**

* Provides all the features similar to collections provided by System.Collections
* Makes the collection type safe
* System.Collections.Generic namespace provides several classes for storing data.
* **List**
* **Stack**
* **Queue**
* **Dictionary**
* **SortedDictionary**

**List**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form6 : Form

{

List<int> list;

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

list = new List<int>(5);

list.Add(10);

list.Add(20);

list.Add(30);

list.Add(40);

list.Add(50);

//list.Remove(30);

//list.RemoveAt(3);

//list.RemoveRange(0, 2);

foreach (int item in list)

{

listBox1.Items.Add(item);

}

}

}

}

**Stack**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form6 : Form

{

Stack<int> stack;

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

stack = new Stack<int>();

stack.Push(10);

stack.Push(20);

stack.Push(30);

stack.Push(40);

stack.Push(50);

stack.Peek();

stack.Pop();

foreach (int item in stack)

{

listBox1.Items.Add(item);

}

}

}

}

**Queue**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form6 : Form

{

Queue<int> queue;

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

queue = new Queue<int>();

queue.Enqueue(10);

queue.Enqueue(20);

queue.Enqueue(30);

queue.Enqueue(40);

queue.Enqueue(50);

queue.Dequeue();

int first = queue.Peek();

foreach (int item in queue)

{

listBox1.Items.Add(item);

}

}

}

}

**Dictionary**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form6 : Form

{

Dictionary<int, string> list;

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

list = new Dictionary<int, string>();

list.Add(101, "value1");

list.Add(234, "abcd");

list.Add(2345, "value2");

list.Add(1234, "cd");

list.Add(124, "xyz");

list.Remove(234);

foreach (KeyValuePair<int, string> item in list)

{

listBox1.Items.Add(item.Key + " : " + item.Value);

}

}

}

}

**SortedDictionary**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form6 : Form

{

SortedDictionary<int, string> list;

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

list = new SortedDictionary<int, string>();

list.Add(101, "value1");

list.Add(234, "abcd");

list.Add(2345, "value2");

list.Add(1234, "cd");

list.Add(124, "xyz");

list.Remove(234);

foreach (KeyValuePair<int, string> item in list)

{

listBox1.Items.Add(item.Key + " : " + item.Value);

}

}

}

}

**File Handling**

* Allows to store / retrieve data on permanent storage
* Files and its data can be handled programmatically
* When a file is opened for reading or writing, it becomes a stream
* System.10 namespace has various classes, used for performing operations with files

using System;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApp1

{

public partial class Form7 : Form

{

public Form7()

{

InitializeComponent();

}

private void Form7\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

//File.Create(@"D:\data\sample.txt");

//File.Move(@"D:\data\sample.txt", @"D:\data\renamedsample.txt");

//File.Delete(@"D:\data\renamedsample.txt");

FileInfo file = new FileInfo(@"D:\data\sampel.txt");

//file.Create();

file.Delete();

}

}

}

**Write & Read in File**

using System;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApp1

{

public partial class Form7 : Form

{

public Form7()

{

InitializeComponent();

}

private void Form7\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

FileInfo file = new FileInfo(@"D:\data\myText.txt");

StreamWriter writer = file.CreateText();

writer.WriteLine(textBox1.Text);

writer.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

FileInfo file = new FileInfo(@"D:\data\myText.txt");

if (file.Exists)

{

StreamReader reader = file.OpenText();

string str = "";

while ((str = reader.ReadLine()) != null)

{

textBox1.Text += str;

}

reader.Close();

}

}

}

}

**Attributes**

* Is a declarative tag used to pass information about the behaviors of various elements at runtime
* A declarative tag is depicted by square ([]) brackets placed above the element it is used for
* .NET provides numerous predefined attributes while developer can also create custom attributes
* All attributes are derived from System.Attributes

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

[AttributeUsage(AttributeTargets.Class)]

class SampleAttribute : Attribute

{

public int Id { get; set; }

public string Name { get; set; }

}

[Sample(Id = 10, Name = "Hello")]

class MyClass

{

[Sample]

public int MyProperty { get; set; }

[Sample]

public void Method()

{

}

}

}

**Extension Methods**

* Enable you to add methods to existing types
* Extension methods are a special kind of static method
* They must exist in a static class

**Class 1**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

int num1 = 235226;

int result = num1.GetDigit();

Console.WriteLine("Dogits :" + result);

}

}

}

**Class 2**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

static class Class2

{

public static int GetDigit(this int num)

{

int count = 0;

while(num != 0)

{

count++;

num = num / 10;

}

return count;

}

}

}

**Assembly**

* Is .NET term for a deployment and configuration unit
* Files with extensions .EXE/.DLL are known as assembly
* Assemblies include metadata
* Are the answer to DLL hell problem
* Can be classified as private and shared assembly

**Private Assembly**

* For simple apps, using private assemblies is the best way.
* Special management, registration, versioning, and so on is not required
* Other applications are not influenced

**Shared Assembly**

* Several applications can use the same assembly
* Reduce the need for disk and memory space
* A shared assembly must have a version number and a unique strong name
* It's installed in the global assembly cache (GAC)

**Assembly Structure**

|  |
| --- |
| MSIL Code |
| Windows Header |
| CLR Header |
| Metadata |
| Manifest |
| Resources |

**MSIL Code:** Is a CPU and platform-agnostic intermediate language. At runtime, MSIL is compiled using JIT compiler.

**Windows Header:** Determines how the Windows operating systems can load and manipulate an assembly. The headers also identify the kind of application such as .dll, console or GUl applications.

**CLR Header:** Is a block of data that all NET assemblies must support in order to be hosted by the CLR.

**Metadata:** Describes the format of the contained types, as well as the format of external type references

**Resources:** NET assembly contains number of embedded resources, such as picture files, application icons, sound file and culture information.

**ManageUser**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ManageUsers

{

public class User

{

public static string HelloUser(string user)

{

return "Hello" + user;

}

public static bool AuthenticateUser(string user, string password)

{

if (user == "akash" && password == "arrow")

return true;

else

return false;

}

}

}

**Form**

using ManageUser;

using System;

using System.Windows.Forms;

using ManageUser;

namespace WindowsFormsApp1

{

public partial class Form8 : Form

{

public Form8()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

User user = new User();

if (user.AuthenticateUser(textBox1.Text, textBox2.Text))

MessageBox.Show(user.HelloUser(textBox1.Text));

else

MessageBox.Show("Invalid Username / Password");

}

}

}

**Reflection**

* It is used for obtaining type information at runtime
* System.Reflection namespace contains classes that allow you to obtain information about the application and to dynamically add types, values, and objects to the application.

**Multi-Threading**

* A thread is defined as the execution path of a program
* Each thread defines a unique flow of control
* Through multithreading, implementation of concurrent programming can be done
* Saves wastage of CPU cycle and increase efficiency
* System.Threading namespace provides threading classes

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

Thread thread1 = new Thread(new ThreadStart(Method1));

Thread thread2 = new Thread(new ThreadStart(Method2));

//thread1.Priority = ThreadPriority.Highest;

thread1.Start();

thread2.Start();

Console.WriteLine("Main thread executes here"); //main thread

}

static void Method1()

{

for (int i = 1; i <= 10; i++) //thread1 executes this

{

if (i == 6)

//Thread.CurrentThread.Abort();

Thread.Sleep(3000);

Console.WriteLine("Method1 : " + i);

}

}

static void Method2()

{

for (int i = 1; i <= 15; i++) //thread2 executes this

{

Console.WriteLine("Method2 : " + i);

}

}

}

}

**Task Parallel Library**

* It makes developers more productive by simplifying the process of adding parallelism and concurrency to applications
* TPL is based on the concept of a task, which represents an asynchronous operation
* Use multiple cores of processor for running multiple tasks parallel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

var t1 = new Task(() => DoWork(1, 1000));

t1.Start();

var t2 = new Task(() => DoWork(2, 3000));

t2.Start();

var t3 = new Task(() => DoWork(3, 1500));

t3.Start();

Console.WriteLine("Pres anhy key to exit..!!");

/\*Task t1 = Task.Factory.StartNew(() => DoWork(1, 2000));

Task t2 = Task.Factory.StartNew(() => DoWork(2, 2500));

Task t3 = Task.Factory.StartNew(() => DoWork(3, 1500));\*/

Console.ReadKey();

}

static void DoWork(int id, int sleep)

{

Console.WriteLine("Task {0} is beginning...", id);

Thread.Sleep(sleep);

Console.WriteLine("Task {0} is completed...", id);

}

}

}

**Asynchronous Programing**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Threading;

namespace WindowsFormsApp1

{

public partial class Form9 : Form

{

public Form9()

{

InitializeComponent();

}

private void Form9\_Load(object sender, EventArgs e)

{

static void BigTask()

{

Thread.Sleep(5000);

}

}

private async void button1\_Click(object sender, EventArgs e)

{

await Task.Run(new Action(BigTask));

label1.Text = "Task Done";

}

private void button2\_Click(object sender, EventArgs e)

{

MessageBox.Show("Test");

}

}

}