**// Experiment no-10**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net.NetworkInformation;

class NetDetails

{

static void Main(string[] args)

{

NetworkInterface[] niArr = NetworkInterface.GetAllNetworkInterfaces();

Console.WriteLine("Retriving basic information of network.\n\n");

foreach (NetworkInterface tempNetworkInterface in niArr)

{

Console.WriteLine("Network Discription : " + tempNetworkInterface.Description);

Console.WriteLine("Network ID : " + tempNetworkInterface.Id);

Console.WriteLine("Network Name : " + tempNetworkInterface.Name);

Console.WriteLine("Network interface type : " + tempNetworkInterface.NetworkInterfaceType.ToString());

Console.WriteLine("Network Operational Status : " + tempNetworkInterface.OperationalStatus.ToString());

Console.WriteLine("Network Spped : " + tempNetworkInterface.Speed);

Console.WriteLine("Support Multicast : " + tempNetworkInterface.SupportsMulticast);

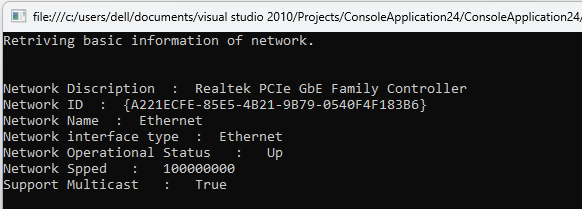
Console.WriteLine();

Console.ReadKey();

}

}

}



using System;

using System.Net.NetworkInformation;

class netchange

{

static void Main(string[] args)

{

// Add the handlers to the NetworkChange events.

NetworkChange.NetworkAvailabilityChanged += NetworkAvailabilityChanged;

NetworkChange.NetworkAddressChanged += NetworkAddressChanged;

Console.WriteLine("Monitoring network changes. Press Enter to exit.");

// Keep the application running until the user decides to exit.

while (Console.ReadLine() != "exit") { }

// Remove the handlers before exiting.

NetworkChange.NetworkAvailabilityChanged -= NetworkAvailabilityChanged;

NetworkChange.NetworkAddressChanged -= NetworkAddressChanged;

}

// Declare a method to handle NetworkAvailabilityChanged events.

private static void NetworkAvailabilityChanged(object sender, NetworkAvailabilityEventArgs e)

{

// Report whether the network is now available or unavailable.

if (e.IsAvailable)

{

Console.WriteLine("Network Available");

}

else

{

Console.WriteLine("Network Unavailable");

}

}

// Declare a method to handle NetworkAddressChanged events.

private static void NetworkAddressChanged(object sender, EventArgs e)

{

Console.WriteLine("Current IP Addresses:");

// Iterate through the interfaces and display information.

foreach (NetworkInterface ni in NetworkInterface.GetAllNetworkInterfaces())

{

foreach (UnicastIPAddressInformation addr in ni.GetIPProperties().UnicastAddresses)

{

Console.WriteLine(" - {0} (lease expires {1})", addr.Address, DateTime.Now.AddSeconds(addr.DhcpLeaseLifetime));

}

}

}

}

